DESAIN & ANALISIS

Sistem Berorientasi Obyek dengan UML

Dr. Joseph Teguh Santoso, S.Kom., M.Kom.
Migunani, M.Kom.

YAYASAN PRIMA AGUS TEKNIK

DESAIN & ANALISIS

Sistem Berorientasi Obyek dengan UML

Dr. Joseph Teguh Santoso, S.Kom., M.Kom.
Migunani, M.Kom.

ASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
JL. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

DESAIN & ANALISIS, Sistem Berorientasi Obyek dengan UML

Penulis :
Dr. Joseph Teguh Santoso, S.Kom., M.Kom.
Migunani, M.Kom.

ISBN : 9 786236 141731

Editor :
Muhammad Sholikhan, M.Kom

Penyunting :
Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

Desain Sampul dan Tata Letak :
Irdha Yunianto, S.Ds., M.Kom.

Penebit :
Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Redaksi :
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :
Universitas STEKOM
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : info@stekom.ac.id

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT, yang telah memberikan kekuatan, ketekunan dan kesabaran sehingga buku yang sudah dipersiapkan ini akhirnya dapat diselesaikan.

Buku ini dipersiapkan terutama untuk mahasiswa ilmu komputer khususnya teknik informatika dan sistem informasi yang sedang mempelajari analisa dan desain sistem. Bahan bacaan analisa dan desain sistem ini sangat dibutuhkan ketika mahasiswa belajar bagaimana tahapan dalam menganalisa dan mendesain suatu sistem atau perangkat lunak berbasis komputer dari tahap awal perencanaan kebutuhan sistem sampai tahap instalasi dan pengoperasian sistem.

Buku ini terdiri dari tiga belas bagian, bab pertama berisi pengantar Pendahuluan Analisis Dan Desain Sistem, bab kedua mengenai Manajemen Proyek, bab ketiga tentang Penentuan Persyaratan Sistem, bab-bab selanjutnya mengenai Proses Bisnis Dan Permodelan Fungsional, Pemodelan Struktural, Pemodelan Perilaku, Pemodelan Desain, Desain Kelas dan Metode, Desain lapisan Manajemen Data, Desain Lapisan Interaksi Manusia Komputer, Desain Lapisan Arsitektur Fisik, Konstruksi, Instalasi dan Operasi Sistem. Setelah mempelajari buku ini diharapkan pembaca khususnya mahasiswa memahami dan dapat menerapkan seluruh tahapan dalam perancangan sistem sampai sistem dapat di terapkan.

Penulis mengucapkan terimakasih kepada berbagai pihak yang telah membantu sehingga dapat diterbitkannya buku ini. penulis juga merasa bahwa buku ini jauh dari sempurna, oleh karena itu segala masukan baik berupa saran maupun kritik yang membangun sangat diharapkan.

Akhirnya semoga buku ini dapat bermanfaat bagi siapa saja yang ingin belajar dan mendalami analisa dan desain sistem berbasis objek.

Semarang, 30 Juli 2021

DAFTAR ISI

BAB 1 PENDAHULUAN ANALISIS DAN DESAIN SISTEM 1

1.1 Tujuan 1

1.2 Pendahuluan 1

1.3 Siklus Hidup Pengembangan Sistem 2

1.4 Metodologi Pengembangan Sistem 5

1.5 Peran Dan Keterampilan Analis Sistem Khusus 19

1.6 Karakteristik Dasar Sistem Berorientasi Objek 21

1.7 Analisis Dan Desain Sistem Berorientasi Obyek/Object-Oriented Systems Analysis And Design (Oosad) 25

1.8 Proses Terpadu (Unified Process) 28

1.9 Unified Modeling Language/Bahasa Pemodelan Terpadu 36

1.10 Menerapkan Konsep Pada Patterson Superstore 39

BAB 2 MANAJEMEN PROYEK 44

2.1 Tujuan 44

2.2 Pendahuluan 44

2.3 Identifikasi Proyek 46

2.4 Analisis Kelayakan 48

2.5 Pilihan Proyek 57

2.6 Peralatan Manajemen Proyek Tradisional 58

2.7 Estimasi Upaya Proyek 63

2.8 Membuat Dan Mengelola Rencana Kerja 58

2.8 Mengatur Staff Proyek 79

2.9 Manajemen Lingkungan Dan Infrastruktur 85

BAB 3 PENENTUAN PERSYARATAN 93

3.1 Tujuan 93

3.2 Pendahuluan 93

3.3 Penentuan Persyaratan 93

3.4 Strategi Analisis Persyaratan 94

3.5 Persyaratan-Teknik Pengumpulan 99

3.6 Teknik Dokumentasi Persyaratan Alternatif 120

3.7 Proposal Sistem 122

BAB 4 PROSES BISNIS DAN PERMODELAN FUNGSIONAL 129

4.1 Tujuan 129

4.2 Pendahuluan 129

4.3 Identifikasi Proses Bisnis Dengan Use-Case Dan Diagram Use-Case 130

4.4 Pemodelan Proses Bisnis Dengan Diagram Aktivitas 139

4.5 Dokumentasi Proses Bisnis Dengan Use-Case Dan Deskripsi Use-Case 149

4.6 Verifikasi Dan Validasi Proses Bisnis Dan Model Fungsional 163

BAB 5 PERMODELAN STRUKTURAL 173

5.1 Tujuan 173

5.2 Pendahuluan 173

5.3 Model Struktur 174

5.4 Identifikasi Obyek 176

5.5 Kartu Crc 182

5.6 Diagram Kelas 186

5.7 Membuat Model Struktur Menggunakan Kartu Crc Dan Diagram Kelas 195

5.8 Verifikasi Dan Validasi Model Struktur 203

BAB 6 PERMODELAN PERILAKU 212

6.1 Tujuan 212

6.2 Pendahuluan 212

6.3 Model Perilaku 213

6.4 Diagram Interaksi 213

6.5 Mesin Status Perilaku 232

6.6 Analisis Crude 239

6.7 Verifikasi Dan Validasi Model Perilaku 242

BAB 7 PINDAH KE DESAIN 248

7.1 Tujuan 248

7.2 Pendahuluan 248

7.3 Mengembangkan Model Analisis Menjadi Model Desain 262

7.4 Paket Dan Diagram Paket 268

7.5 Strategi Desain 274

7.6 Memilih Strategi Akuisisi 280

BAB 8 DESAIN KELAS DAN METODE 286

8.1 Tujuan 286

8.2 Pendahuluan 286

8.3 Tinjauan Karakteristik Dasar Orientasi Obyek 288

8.4 Kriteria Desain 293

8.5 Kegiatan Desain Objek 300

8.6 Kendala Dan Kontrak 311

8.7 Verifikasi Dan Validasi Desain Kelas Dan Metode 327

BAB 9 DESAIN LAPISAN MANAJEMEN DATA 334

9.1 Tujuan : 334

9.2 Pendahuluan 334

9.3 Format Persisten Objek..... 335

9.4 Masalah Pemetaan Objek Domain Ke Format Persistensi Objek 345

9.5 Mengoptimalkan Penyimpanan Obyek Berbasis Rdbmsms 355

9.6 Merancang Kelas Akses Data Dan Manipulasi 367

9.7 Persyaratan Non Fungsional Dan Desain Lapisan Manajemen Data..... 369

9.8 Verifikasi Dan Validasi Lapisan Manajemen Data..... 371

BAB 10 DESAIN LAPISAN INTERAKSI MANUSIA-KOMPUTER..... 376

10.1 Tujuan 376

10.2 Pendahuluan 376

10.3 Prinsip Untuk Desain Antarmuka Pengguna (User Interface/Ui) 377

10.4 Proses Desain Antarmuka Pengguna/ User Interface (Ui)..... 381

10.5 Desain Navigasi 394

10.6 Input Desain 400

10.7 Output Desain 407

10.8 Komputasi Seluler Dan Desain Antarmuka Pengguna 411

10.9 Media Sosial Dan Desain Antarmuka Pengguna..... 414

10.10 Game, Visualisasi Informasi Multidimen, Dan Lingkungan Yang Menakjubkan 416

10.11 Isu Internasional Dan Budaya Dan Desain Antarmuka Pengguna 423

10.12 Persyaratan Non Fungsional Dan Desain Lapisan Interaksi Manusia-Komputer 427

10.13 Minicase..... 431

BAB 11 DESAIN LAPISAN ARSITEKTUR FISIK 437

11.1 Tujuan 437

11.2 Pendahuluan 437

11.3 Elemen Lapisan Arsitektur Fisik 438

11.4 Komputasi Cloud 445

11.5 Komputasi Di Mana-Mana Dan Internet Of Thing 448

11.6 Green It 451

11.7 Desain Infrastruktur 452

11.8 Spesifikasi Perangkat Keras Dan Perangkat Lunak Sistem 457

11.9 Persyaratan Nonfungsional Dan Desain Lapisan Arsitektur Fisik 460

11.10 Verifikasi Dan Validasi Lapisan Arsitektur Fisik 469

BAB 12 KONSTRUKSI 474

12.1 Tujuan 474

12.2 Pendahuluan 474

12.3 Mengelola Pemrograman 475

12.4 Mengembangkan Dokumentasi 480

12.5 Tes Perancangan 487

BAB 13 INSTALASI DAN OPERASI 503

13.1 Tujuan 503

13.2 Pendahuluan 503

13.3 Isu Budaya Dan Adopsi Teknologi Informasi 505

13.4 Konversi 507

13.5 Perubahan Manajemen 512

13.6 Kegiatan Pasca Pelaksanaan 521

BAB 1

PENDAHULUAN ANALISIS DAN DESAIN SISTEM

Pada bab ini memperkenalkan siklus hidup pengembangan sistem (systems development life cycle/SDLC), model empat fase dasar (perencanaan, analisis, desain, dan implementasi) yang umum untuk semua proyek pengembangan sistem informasi. Bab ini menggambarkan mengenai evolusi metodologi pengembangan sistem dan membahas tugas dan kemampuan yang dibutuhkan oleh seorang analis sistem. Bab ini kemudian mengulas karakteristik dasar dari suatu sistem yang berorientasi objek dan dasar-dasar analisis sistem dan desain yang berorientasi objek dan ditutup dengan deskripsi Proses Terpadu (Unified Process) dan ekstensinya serta Bahasa Pemodelan Terpadu (Unified Modeling Language).

1.1 TUJUAN

- Memahami siklus hidup pengembangan sistem dasar dan empat fasenya
- Memahami evolusi metodologi pengembangan sistem
- Kenali berbagai tugas dan keterampilan seorang analis sistem
- Kenali karakteristik dasar sistem yang berorientasi objek
- Kenali prinsip-prinsip dasar analisis dan desain sistem berorientasi objek
- Pahami Proses Terpadu, ekstensinya, dan Bahasa Pemodelan Terpadu

1.2 PENDAHULUAN

Siklus hidup pengembangan sistem (systems development life cyle/SDLC) adalah proses memahami bagaimana sistem informasi (SI) dapat mendukung persyaratan bisnis dengan cara merancang sistem, membangunnya, dan mengirimkannya kepada pengguna. Jika Anda telah mengambil kelas pemrograman atau telah memprogram sendiri, hal ini mungkin terdengar cukup sederhana. Sebuah survei tahun 1996 yang dilakukan oleh Standish Group menemukan bahwa 42 persen dari semua proyek SI perusahaan berhenti sebelum selesai. Sebuah studi serupa yang dilakukan pada tahun 1996 oleh Kantor Akuntansi Umum (General Accounting Office) menemukan bahwa 53 persen dari semua proyek SI pemerintah ditinggalkan. Sayangnya, banyak sistem yang diselesaikan dikirim ke pengguna tetapi terlambat secara signifikan, jauh lebih mahal dari yang direncanakan, dan memiliki lebih sedikit fitur daripada yang semula direncanakan. Sebagai contoh, IAG Consulting melaporkan bahwa 80 persen dari proyek-proyek itu terlambat, 72 persen melebihi anggaran, dan 55 persen tidak berfungsi dengan baik; Panorama Consulting Solutions melaporkan bahwa 54 persen proyek ERP melebihi batas waktu, 56 persen melebihi anggaran, dan 48 persen memberikan kurang dari 50 persen manfaat sebagaimana yang direncanakan; dan sebuah penelitian yang dilakukan oleh IBM melaporkan bahwa 59 persen proyek tidak tepat waktu, tidak sesuai anggaran, dan terkendala kualitas.[1] Meskipun kami ingin mempromosikan buku ini sebagai peluru perak yang akan menjauhkan Anda dari kegagalan SI, kami mengakui bahwa sebenarnya peluru perak yang menjamin keberhasilan pengembangan SI sama sekali

[1] Untuk informasi lebih lanjut tentang masalah ini, lihat Capers Jones, Patterns of Software System Future and Succes (London: International Thompson Computer Press, 1996); KeithEllis, Business Analysis Benchmark: The Impact of Business Requirements on the Succes of Technology Projects (2008). Diperoleh pada Mei 2014 dari IAG Consulting, www.iag.biz; H. H. Jorgensen, L. Owen, dan A. Neus, Making Change Work (2008). Diperoleh pada Mei 2014 dari IBM, www.ibm. com; Panorama Consulting Solutions, 2012 ERP Report (2012). Diperoleh pada Mei 2014 dari Panorama-Consulting.com.

tidak ada. Sebaliknya, buku ini memberi Anda beberapa konsep dasar dan beberapa teknik praktis yang dapat Anda gunakan untuk meningkatkan peluang untuk berhasil.

Orang yang menjadi utama dalam SDLC adalah analis sistem, yang menganalisis situasi bisnis, mengidentifikasi peluang untuk peningkatan, dan merancang sistem informasi untuk mengimplementasikannya. Menjadi seorang analis sistem adalah salah satu pekerjaan yang paling menarik, mengasyikkan, dan menantang. Analis sistem bekerja dengan berbagai jenis orang dan belajar bagaimana mereka menjalankan bisnisnya. Secara khusus, mereka bekerja dengan tim analis sistem, programmer, dll dalam misi yang sama. Analis sistem merasakan kepuasan melihat sistem yang mereka rancang dan kembangkan dapat memberikan dampak bisnis yang signifikan dan mengetahui bahwa mereka berkontribusi dalam memberikan keterampilan unik mereka untuk mewujudkannya.

Namun, tujuan utama seorang analis sistem bukanlah menciptakan sistem yang luar biasa; tetapi untuk menciptakan nilai bagi organisasi, yang bagi sebagian besar perusahaan artinya meningkatkan laba (lembaga pemerintah dan organisasi nirlaba mengukur nilai secara berbeda). Banyak sistem gagal yang ditinggalkan karena para analis mencoba membangun sistem yang bagus tanpa memahami dengan jelas kesesuaian sistem dengan tujuan organisasi, kesesuaian dengan proses bisnis yang berjalan saat ini, dan kesesuaian dengan sistem informasi lainnya. Investasi dalam sistem informasi sama seperti investasi lainnya. Tujuannya bukan untuk memperoleh alat, karena alat itu hanyalah sarana untuk mencapai tujuan; tujuannya adalah untuk memungkinkan organisasi melakukan pekerjaan dengan lebih baik sehingga dapat memperoleh keuntungan yang lebih besar atau melayani konstituennya secara lebih efektif.

Buku ini memperkenalkan keterampilan dasar yang dibutuhkan oleh seorang analis sistem. Buku pragmatis ini membahas praktik terbaik dalam pengembangan sistem; buku ini tidak menyajikan survei umum pengembangan sistem yang mencakup segala sesuatu tentang topik tersebut. Berdasarkan definisi, analis sistem bertugas melakukan berbagai hal dan menguji cara kerja organisasi. Untuk mendapatkan hasil maksimal dari buku ini, Anda perlu secara aktif menerapkan ide-ide dan konsep-konsep dalam contoh proyek pengembangan sistem Anda sendiri. Buku ini memandu Anda melewati setiap tahap untuk menghasilkan sistem informasi yang bagus. Pada saat Anda menyelesaikan buku ini, Anda tidak akan menjadi analis ahli, tetapi Anda akan siap untuk mulai membangun sistem secara nyata.

1.3 SIKLUS HIDUP PENGEMBANGAN SISTEM

Dalam banyak hal, membangun sistem informasi mirip dengan membangun rumah. Pertama, rumah (atau sistem informasi) dimulai dengan ide dasar. Kedua, ide ini diubah menjadi gambar sederhana yang diperlihatkan kepada pelanggan dan disempurnakan (seringkali melalui beberapa gambar, dan beberapa revisi) sampai pelanggan setuju bahwa gambar tersebut menggambarkan apa yang dia inginkan. Ketiga, satu set blueprint dirancang dengan menyajikan informasi yang jauh lebih rinci tentang rumah (misalnya, jenis keran air atau di mana kabel jack telepon). Terakhir, rumah dibangun mengikuti blueprint, seringkali dengan beberapa perubahan yang diminta oleh pelanggan saat rumah didirikan.

SDLC memiliki rangkaian empat fase mendasar yang serupa: perencanaan, analisis, desain, dan implementasi. Proyek yang berbeda mungkin memiliki penekanan SDLC yang berbeda atau memilik pendekatan fase SDLC dengan cara yang berbeda, tetapi semua proyek memiliki elemen dari empat fase ini. Setiap fase itu sendiri terdiri dari serangkaian langkah, yang tergantung pada teknik yang menghasilkan produk yang dapat dikirim (dokumen dan file spesifik yang menyediakan informasi mengenai pemahaman tentang proyek).

Misalnya, dalam mendaftar ke universitas, semua mahasiswa melalui fase yang sama: mengumpulkan informasi, mendaftar, dan diterima. Masing-masing fase ini memiliki langkah; misalnya, pengumpulan informasi mencakup langkah-langkah seperti mencari universitas, meminta informasi, dan membaca brosur. Mahasiswa kemudian menggunakan teknik (misalnya, pencarian melalui internet) yang dapat diterapkan pada langkah-langkah (misalnya, meminta informasi) untuk membuat kiriman (misalnya, evaluasi berbagai aspek universitas).

Dalam banyak proyek, fase dan langkah SDLC berlangsung melalui jalur yang logis dari awal hingga akhir. Dalam proyek lain, tim proyek bergerak melalui langkah-langkah secara berurutan, bertahap, berulang, atau melalui pola lain. Di bagian ini, kami menjelaskan fase, tindakan, dan beberapa teknik yang digunakan untuk menyelesaikan langkah pada tingkat yang sangat tinggi.

Untuk saat ini, ada dua poin penting yang harus dipahami tentang SDLC. Pertama, Anda harus memahami secara umum fase dan langkah yang dilalui proyek SI dan beberapa teknik yang menghasilkan kiriman tertentu. Kedua, penting untuk dipahami bahwa SDLC adalah proses penyempurnaan bertahap. Kiriman yang dihasilkan dalam fase analisis memberikan gambaran umum tentang bentuk sistem baru. Hasil ini digunakan sebagai masukan untuk fase desain, yang kemudian digunakan untuk menyempurnakannya dan untuk menghasilkan satu set kiriman yang dapat menjelaskan dalam istilah yang jauh lebih rinci secara persis mengenai bagaimana sistem akan dibangun. Hasil ini, pada gilirannya, digunakan dalam fase implementasi untuk menghasilkan sistem yang sebenarnya. Setiap fase menyempurnakan dan mengelaborasi pekerjaan yang dilakukan pada fase sebelumnya.

Perencanaan

Fase perencanaan adalah proses mendasar untuk memahami mengapa sistem informasi harus dibangun dan menentukan bagaimana tim proyek akan membangunnya. Perencanaan ini memiliki dua langkah:

1. Selama inisialisasi proyek, nilai bisnis sistem terhadap organisasi diidentifikasi: Bagaimana cara menurunkan biaya atau meningkatkan pendapatan? Sebagian besar ide untuk sistem baru datang dari luar area SI (misalnya, dari departemen pemasaran, departemen akuntansi) dalam bentuk permintaan sistem. Permintaan sistem menyajikan ringkasan singkat mengenai persyaratan bisnis, dan menjelaskan bagaimana sistem yang mendukung persyaratan dapat menciptakan nilai bisnis. Departemen SI bekerja sama dengan orang atau departemen yang mengusulkan permintaan (disebut sponsor proyek) untuk melakukan analisis kelayakan.

   Permintaan sistem dan analisis kelayakan ditunjukkan kepada komite persetujuan sistem informasi (kadang-kadang disebut komite pengarah), yang memutuskan apakah proyek harus dilakukan.

2. Setelah proyek disetujui, lalu masuk ke manajemen proyek. Selama manajemen proyek, manajer proyek membuat rencana kerja, staf proyek, dan menerapkan teknik untuk membantu tim proyek mengendalikan dan mengarahkan proyek melalui seluruh tahapan SDLC. Kiriman untuk manajemen proyek adalah rencana proyek, yang menjelaskan bagaimana tim proyek akan mengembangkan sistem.

Analisis

Tahap analisis menjawab pertanyaan tentang siapa yang akan menggunakan sistem, apa yang akan dilakukan sistem, dan di mana dan kapan akan digunakan. Selama fase ini, tim proyek menyelidiki setiap sistem yang digunakan saat ini, mengidentifikasi peluang untuk perbaikan, dan mengembangkan konsep untuk sistem baru.

Fase ini memiliki tiga langkah:

1. Strategi analisis dikembangkan untuk memandu upaya tim proyek. Strategi itu biasanya mencakup analisis sistem saat ini (disebut sistem apa adanya) dan berbagai permasalahnya dan cara merancang sistem baru (disebut sistem yang akan datang).
2. Langkah selanjutnya adalah pengumpulan persyaratan (misalnya, melalui wawancara atau kuesioner). Analisis informasi ini—dalam hubungannya dengan masukan dari sponsor proyek dan orang lain—mengarah pada pengembangan konsep untuk sistem baru. Konsep sistem tersebut kemudian digunakan sebagai dasar untuk mengembangkan seperangkat model analisis bisnis, yang menggambarkan bagaimana bisnis akan beroperasi jika sistem baru dikembangkan.
3. Analisis, konsep sistem, dan model digabungkan ke dalam dokumen yang disebut proposal sistem, yang dipresentasikan kepada sponsor proyek dan pengambil keputusan utama lainnya (misalnya, anggota komite persetujuan) yang memutuskan apakah proyek harus terus berlanjut.

Proposal sistem adalah kiriman awal yang menjelaskan persyaratan bisnis apa yang harus dipenuhi oleh sistem baru. Karena ini merupakan langkah pertama dalam desain sistem baru, beberapa ahli berpendapat bahwa tidak tepat untuk menggunakan istilah "analisis" sebagai nama untuk fase ini; mereka berpendapat bahwa nama yang lebih baik adalah "analisis dan desain awal." Namun, sebagian besar organisasi tetap menggunakan analisis sebagai nama untuk fase ini, jadi kami juga menggunakannya dalam buku ini. Perlu diingat bahwa kiriman dari fase analisis adalah analisis dan desain awal tingkat tinggi untuk sistem baru.

Desain

Fase desain memutuskan bagaimana sistem akan beroperasi, dalam hal perangkat keras, perangkat lunak, dan infrastruktur jaringan; antarmuka pengguna, formulir, dan laporan; dan program tertentu, database, dan file yang akan dibutuhkan. Meskipun sebagian besar keputusan strategis tentang sistem dibuat dalam pengembangan konsep sistem selama fase analisis, langkah-langkah dalam fase desain menentukan secara pasti bagaimana sistem akan beroperasi. Fase desain memiliki empat langkah:

1. Pertama, strategi desain pertama dikembangkan. Tahap ini menjelaskan apakah sistem akan dikembangkan oleh pemrogram perusahaan sendiri, ataukah sistem akan dialihkan ke perusahaan lain (biasanya perusahaan konsultan), atau apakah perusahaan akan membeli paket perangkat lunak yang sudah ada.
2. Kemudian langkah berikutnya mengarah pada pengembangan desain arsitektur dasar sistem, yang menggambarkan perangkat keras, perangkat lunak, dan infrastruktur jaringan yang akan digunakan. Dalam kebanyakan kasus, sistem akan menambah atau mengubah infrastruktur yang sudah ada dalam organisasi. Desain antarmuka menentukan bagaimana pengguna akan bergerak melalui sistem (misalnya, metode navigasi seperti menu dan tombol di layar) dan formulir serta laporan yang akan digunakan sistem.
3. Langkah berikutnya, database dan spesifikasi file dikembangkan. Tahap ini mendefinisikan dengan tepat data apa yang akan disimpan dan di mana itu akan disimpan.
4. Tim analis mengembangkan desain program, yang menentukan program yang perlu ditulis dan apa yang akan dilakukan setiap program secara tepat.

Kumpulan kiriman ini (desain arsitektur, desain antarmuka, database dan spesifikasi file, dan desain program) adalah spesifikasi sistem yang diserahkan kepada tim pemrograman untuk diimplementasikan. Pada akhir fase desain, analisis kelayakan dan rencana proyek

diperiksa ulang dan direvisi, dan keputusan lain dibuat oleh sponsor proyek dan komite persetujuan mengenai apakah proyek akan dihentikan atau dilanjutkan.

Implementasi

Fase terakhir dalam SDLC adalah fase implementasi, di mana sistem benar-benar dibangun (atau dibeli, dalam hal desain paket perangkat lunak). Ini adalah fase yang biasanya mendapat perhatian paling besar, karena untuk sebagian besar sistem, fase ini adalah proses pengembangan yang paling lama dan paling mahal. Fase ini memiliki tiga langkah:

1. Konstruksi sistem merupakan langkah pertama. Sistem dibangun dan diuji untuk memastikan bahwa sistem itu bekerja sesuai desain. Karena biaya bug bisa sangat besar, pengujian adalah salah satu langkah paling penting dalam implementasi. Sebagian besar organisasi memberikan lebih banyak waktu dan perhatian untuk pengujian dibanding penulisan program.
2. Kemudian sistem telah terpasang. Instalasi adalah proses di mana sistem lama dimatikan dan yang baru dihidupkan. Salah satu aspek terpenting dari konversi adalah pengembangan rencana pelatihan untuk mengajari pengguna mengenai cara menggunakan sistem baru dan membantu mengelola perubahan yang disebabkan oleh adanya sistem baru.
3. Tim analis menetapkan support plan untuk sistem. Tahap ini biasanya mencakup tinjauan pasca implementasi formal atau informal serta cara sistematis untuk mengidentifikasi perubahan besar dan kecil yang diperlukan untuk sistem.

1.4 METODOLOGI PENGEMBANGAN SISTEM

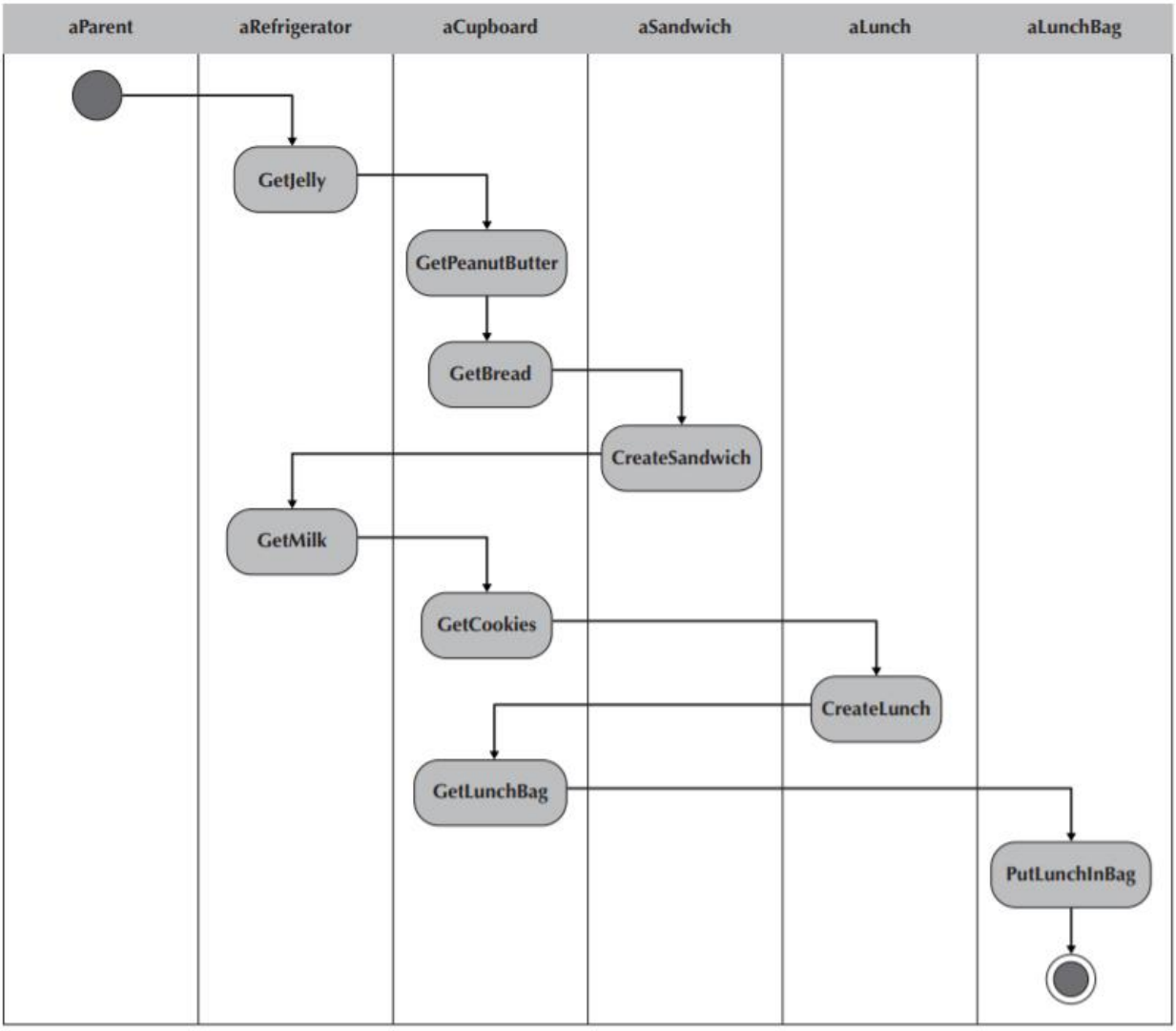
Metodologi adalah pendekatan formal untuk mengimplementasikan SDLC (yaitu, daftar tahapan dan kiriman). Ada banyak metodologi pengembangan sistem yang berbeda, dan masing-masing memiliki ciri khas tersendiri, berdasarkan urutan dan fokus yang ada pada setiap fase SDLC. Beberapa metodologi merupakan standar formal yang digunakan oleh lembaga pemerintahan, sedangkan metode lainnya bisa juga dikembangkan oleh perusahaan konsultan untuk dijual kepada klien. Beberapa organisasi memiliki metodologi internal yang telah diasah selama bertahun-tahun, dan metodologi itu dapat menjelaskan dengan tepat bagaimana setiap fase SDLC harus dilakukan di perusahaan itu.

Ada banyak cara untuk mengkategorikan metodologi. Salah satu caranya adalah dengan melihat apakah metodologi itu fokus pada proses bisnis atau pada data yang mendukung bisnis. Metodologi terpusat proses merupakan metodologi yang menekankan model proses sebagai inti dari konsep sistem. Pada Gambar 1-1, misalnya, metodologi terpusat proses akan memfokuskan pertama kali pada pendefinisian proses (misalnya, menyusun bahan-bahan sandwich). Metodologi terpusat data menekankan model data sebagai inti dari konsep sistem. Pada Gambar 1-1, metodologi terpusat data akan memfokuskanan pertama kali pada pendefinisian isi dari area penyimpanan (misalnya, lemari es) dan bagaimana isinya diatur.[2] Sebaliknya, metodologi yang berorientasi objek berusaha untuk menyeimbangkan fokus antara proses dan data dengan menggabungkan keduanya ke dalam satu model. Pada Gambar 1-1, metodologi ini akan memfokuskan pertama kali pada pendefinisian elemen utama dari

[2] Metodologi terpusat pada proses klasik modern adalah metodologi yang dikemukakan oleh Edward Yourdon, Modern Structured Analysis (Englewood Cliffs, NJ: Yourdon Press, 1989). Contoh metodologi terpusat pada data adalah rekayasa informasi; lihat James Martin, Information Engineering, jilid 1-3 (Englewood Cliffs, NJ: Prentice Hall, 1989). Metodologi standar tak berorientasi pada objek yang diterima secara luas, yang menyeimbangkan proses dan data adalah IDEF; lihat FIPS 183, Integration Definiton for Function Modeling, Federal Information Processing Standards Publications, Departemen Perdagangan AS, 1993.

sistem (misalnya, sandwich, makan siang) dan melihat proses dan data yang terlibat dengan setiap elemen.

Faktor penting lainnya dalam mengkategorikan metodologi adalah urutan fase SDLC dan total waktu dan upaya yang diperlukan untuk masing-masing fase tsb.[3] Pada awal-awal munculnya komputasi, programmer tidak memahami perlunya metodologi siklus hidup yang formal dan terencana. Mereka cenderung bekerja langsung dari fase perencanaan yang sangat sederhana ke langkah konstruksi fase implementasi—dengan kata lain, dari permintaan sistem yang sangat kasar dan yang tidak dipikirkan dengan matang langsung ke tahap penulisan kode. Ini adalah pendekatan yang sama yang terkadang Anda gunakan saat menulis program untuk kelas pemrograman. Cara ini dapat bekerja untuk program kecil yang hanya membutuhkan satu programmer, tetapi jika persyaratannya rumit atau tidak jelas, Anda mungkin kehilangan aspek penting dari permasalahan itu dan harus memulai dari awal lagi, membuang beberapa hal dari program (dan membuang-buang waktu serta upaya untuk menulisnya). Pendekatan ini juga membuat kerja tim menjadi sulit karena anggota memiliki sedikit gagasan tentang apa yang perlu dicapai dan bagaimana bekerja sama untuk menghasilkan produk akhir. Pada bagian ini, kami menjelaskan tiga kelas metodologi pengembangan sistem yang berbeda: desain terstruktur, pengembangan aplikasi cepat, dan pengembangan tangkas.

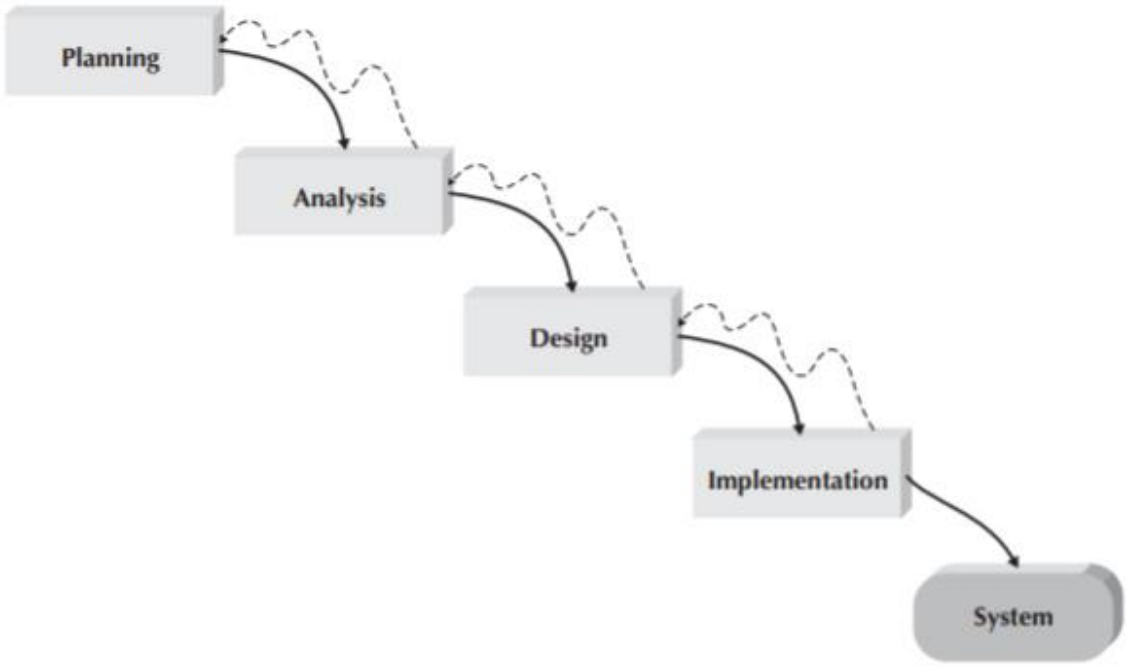


[3] Referensi yang bagus untuk membandingkan metodologi pengembangan sistem adalah Steve McConnell, Rapid Development (Redmond, WA: Microsoft Press, 1996).

GAMBAR 1-1 Model Perilaku Sederhana untuk Membuat Makan Siang Sederhana

Desain Terstruktur

Kategori pertama dari metodologi pengembangan sistem disebut desain terstruktur. Metodologi ini dominan pada tahun 1980-an, menggantikan pendekatan ad hoc yang tidak disiplin sebelumnya. Metodologi desain terstruktur mengadopsi pendekatan langkah demi langkah formal (step-by-step) ke SDLC yang bergerak secara logis dari satu fase ke fase berikutnya. Banyak metodologi terpusat pada proses dan terpusat pada data yang mengikuti pendekatan dasar dari dua kategori desain terstruktur yang akan dibahas nanti.



GAMBAR 1-2 Metodologi Berbasis Pengembangan Waterfall

Pengembangan Waterfall. Metodologi desain terstruktur original (masih digunakan sampai sekarang) adalah pengembangan waterfall. Dengan metodologi berbasis pengembangan waterfall, analis dan pengguna menjalankan secara berurutan proses dari satu fase ke fase berikutnya (lihat Gambar 1-2). Kiriman utama untuk setiap fase biasanya sangat panjang (seringkali panjangnya hingga ratusan halaman) dan dipresentasikan kepada sponsor proyek untuk disetujui saat proyek berjalan dari fase ke fase. Setelah sponsor menyetujui pekerjaan yang dilakukan dalam fase tertentu, fase itu selesai dan fase berikutnya dimulai. Metodologi ini disebut sebagai metodologi pengembangan waterfall karena bergerak maju dari fase ke fase dengan cara yang sama seperti water-fall. Meskipun mungkin untuk mundur dalam SDLC (misalnya, dari desain kembali ke analisis), tetapi hal itu sangat sulit (bayangkan Anda adalah ikan salmon yang mencoba berenang ke hulu melawan water-fall, seperti yang ditunjukkan pada Gambar 1-2).

Desain terstruktur juga memperkenalkan penggunaan pemodelan formal atau teknik diagram untuk menggambarkan proses bisnis dasar dan data yang mendukungnya. Desain terstruktur tradisional menggunakan satu set diagram untuk mewakili proses dan satu set diagram terpisah untuk mewakili data. Karena dua set diagram digunakan, analis sistem harus memutuskan set mana yang akan dikembangkan terlebih dahulu dan digunakan sebagai inti sistem: diagram model proses atau diagram model data.

Dua keuntungan utama dari pendekatan waterfall desain terstruktur adalah bahwa pendekatan itu mengidentifikasi persyaratan sistem jauh sebelum pemrograman dimulai dan

meminimalkan perubahan persyaratan saat proyek berlangsung. Dua kelemahan utama adalah bahwa desain harus benar-benar ditentukan sebelum pemrograman dimulai dan waktu yang dibutuhkan lama antara penyelesaian proposal sistem dalam fase analisis dan pengiriman sistem (biasanya berbulan-bulan atau bertahun-tahun). Jika tim proyek melewatkan persyaratan penting, pemrograman pasca-implementasi yang memerlukan biaya mahal mungkin diperlukan (bayangkan Anda mencoba mendesain mobil di atas kertas; seberapa besar kemungkinan Anda akan mengingat lampu interior yang menyala ketika pintu terbuka atau untuk menentukan jumlah yang tepat dari katup pada mesin?). Sebuah sistem juga dapat memerlukan pengerjaan ulang yang signifikan karena adanya perubahan pada lingkungan bisnis sejak fase analisis dilakukan.

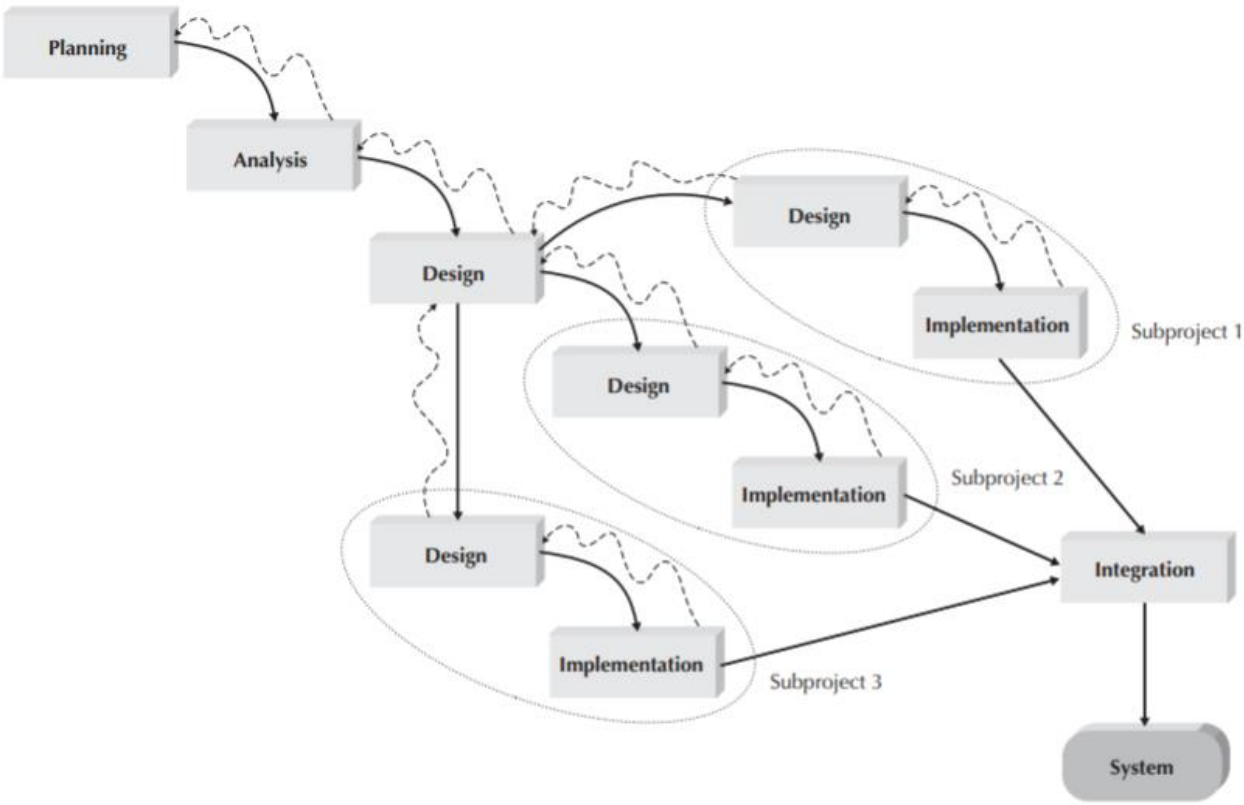
Pengembangan Paralel. Metodologi pengembangan paralel mencoba untuk mengatasi masalah penundaan yang lama antara fase analisis dan pengiriman sistem. Alih-alih melakukan desain dan implementasi secara berurutan, metodologi ini melakukan desain umum untuk keseluruhan sistem dan kemudian membagi proyek menjadi serangkaian subproyek berbeda yang dapat dirancang dan diimplementasikan secara paralel atau beriringan. Setelah semua subproyek selesai, bagian-bagian yang terpisah diintegrasikan dan sistem dikirimkan (lihat Gambar 1-3).

Keuntungan utama dari metodologi ini adalah dapat mengurangi waktu untuk membuat sistem; dengan demikian, kemungkinan perubahan dalam lingkungan bisnis yang menyebabkan pengerjaan ulang dapat dikurangi. Namun, terkadang subproyek tidak sepenuhnya independen; keputusan desain yang dibuat dalam satu subproyek dapat memengaruhi subproyek lainnya, dan pada akhir proyek mungkin dibutuhkan upaya integrasi yang signifikan.

Pengembangan Aplikasi Cepat/Rapid Application Development (RAD)

Kategori metodologi kedua termasuk metodologi berbasis pengembangan aplikasi cepat (rapid application development/RAD). Metodologi ini adalah kelas metodologi pengembangan sistem yang lebih baru yang muncul pada tahun 1990-an. Metodologi berbasis RAD mencoba untuk mengatasi dua kelemahan metodologi desain terstruktur dengan menyesuaikan fase SDLC untuk memperoleh beberapa bagian dari sistem yang dikembangkan dengan cepat dan sampai ke tangan pengguna. Dengan cara ini, pengguna dapat lebih memahami sistem dan menyarankan revisi yang membawa sistem lebih dekat ke apa yang dibutuhkan.[4]

[4] Salah satu buku RAD terbaik adalah Steve McConnell, Rapid Development (Redmond, WA: Microsoft Press, 1996).



GAMBAR 1-3 Metodologi Berbasis Pengembangan Paralel

Sebagian besar metodologi berbasis RAD merekomendasikan agar analis menggunakan teknik khusus dan alat komputer untuk mempercepat fase analisis, desain, dan implementasi, seperti alat computer-aided software engineering (CASE), sesi joint application design (JAD), sesi generasi keempat atau bahasa pemrograman visual yang menyederhanakan dan mempercepat pemrograman, dan pembuat kode yang secara otomatis menghasilkan program dari spesifikasi desain. Kombinasi fase SDLC yang diubah dan penggunaan alat dan teknik ini meningkatkan kecepatan dan kualitas pengembangan sistem. Namun, kemungkinan ada satu masalah kecil dengan metodologi berbasis RAD, yaitu mengelola ekspektasi pengguna. Karena penggunaan alat dan teknik yang dapat meningkatkan kecepatan dan kualitas pengembangan sistem, ekspektasi pengguna tentang apa yang dapat dilakukan dapat berubah secara drastis. Karena pengguna semakin baik dalam memahami teknologi informasi (TI), persyaratan sistem cenderung berkembang. Hal ini tidak terlalu menjadi masalah saat menggunakan metodologi yang menghabiskan banyak waktu untuk mendokumentasikan persyaratan secara menyeluruh.

Pengembangan Bertahap. Sebuah metodologi berbasis pengembangan bertahap memecah keseluruhan sistem menjadi serangkaian versi yang dikembangkan secara berurutan. Fase analisis mengidentifikasi konsep sistem secara keseluruhan, dan tim proyek, pengguna, dan sponsor sistem kemudian mengkategorikan persyaratan ke dalam serangkaian versi. Persyaratan yang paling penting dan mendasar digabungkan ke dalam versi pertama sistem. Fase analisis kemudian mengarah ke desain dan implementasi—tetapi hanya dengan serangkaian persyaratan yang diidentifikasi untuk versi 1 (lihat Gambar 1-4).

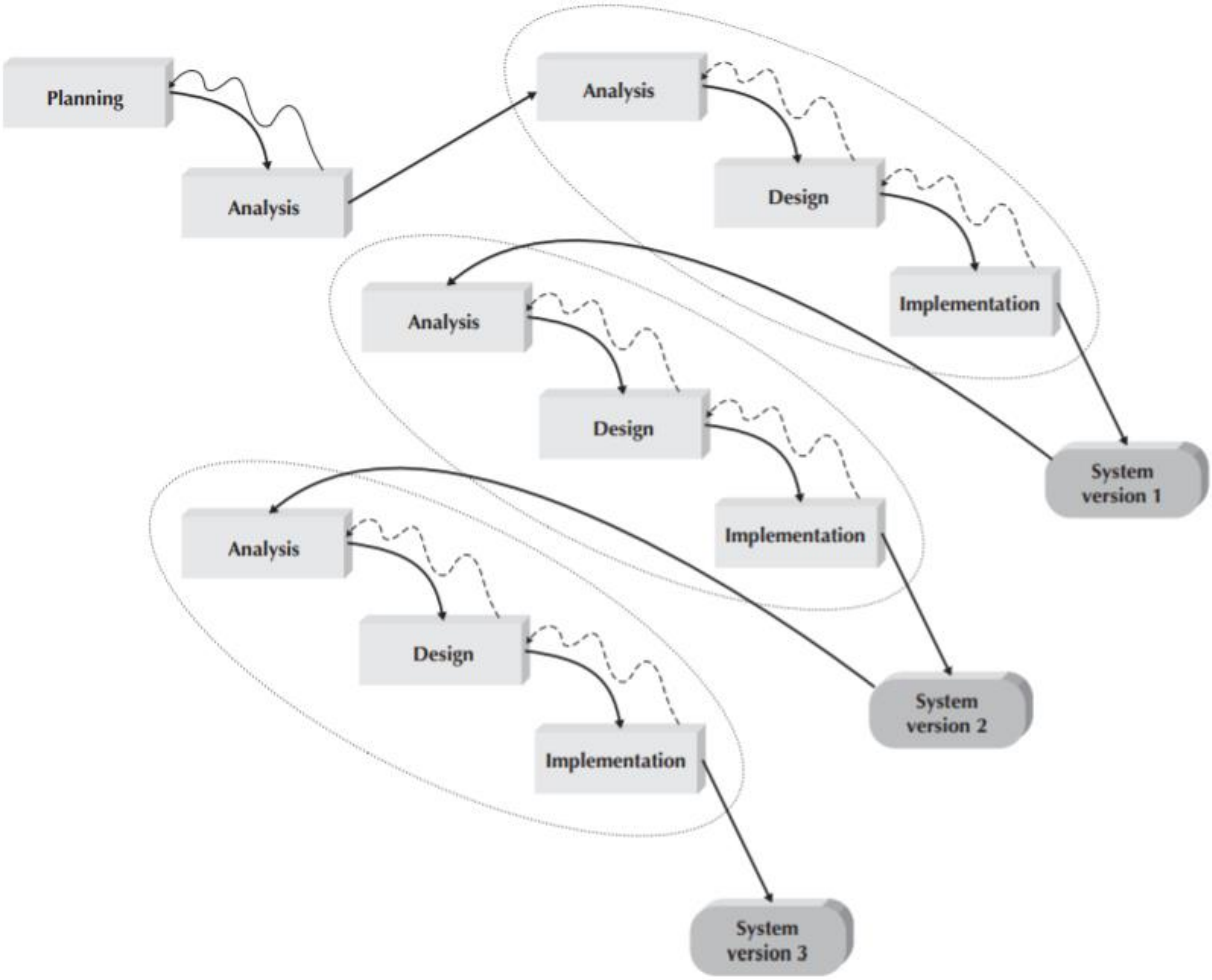
Setelah versi 1 diimplementasikan, kemudian pekerjaan mulai dilakukan pada versi 2. Analisis tambahan dilakukan berdasarkan persyaratan yang telah diidentifikasi sebelumnya dan dikombinasikan dengan ide dan masalah baru yang muncul dari pengalaman pengguna dengan versi 1. Versi 2 kemudian dirancang dan diimplementasikan, dan pekerjaan segera

dimulai untuk versi berikutnya. Proses ini berlanjut sampai sistem selesai atau tidak lagi digunakan.

Metodologi berbasis pengembangan bertahap memiliki keuntungan seperti cepat dalam memperoleh sistem yang berguna bagi pengguna. Meskipun sistem tidak melakukan semua fungsi yang dibutuhkan oleh pengguna di awal, sistem ini dapat memberikan nilai bisnis lebih cepat dibanding jika sistem dikirimkan setelah selesai, seperti halnya dengan metodologi waterfall dan paralel. Demikian juga, karena pengguna mulai bekerja dengan sistem lebih awal, mereka lebih mungkin untuk mengidentifikasi persyaratan tambahan yang penting daripada jika dengan desain terstruktur.

Kelemahan utama pengembangan bertahap adalah bahwa pengguna mulai bekerja dengan sistem tidak lengkap yang memang disengaja. Sangat penting untuk mengidentifikasi fitur yang paling penting dan berguna dan memasukkannya ke dalam versi pertama dan mengelola ekspektasi pengguna di sepanjang perjalanan sistem.

Prototyping. Metodologi berbasis prototyping melakukan fase analisis, desain, dan implementasi secara bersamaan, dan ketiga fase dilakukan berulang kali dalam satu siklus sampai sistem selesai. Dengan metodologi ini, dasar-dasar analisis dan desain dikerjakan, dan pekerjaan segera dimulai pada prototipe sistem, yaitu program cepat dan kotor (quick-and-dirty) yang menyediakan fitur dalam jumlah minimal. Prototipe pertama biasanya merupakan bagian pertama dari sistem yang digunakan. Prototipe ini ditunjukkan kepada pengguna dan sponsor proyek, yang akan memberikan komentar. Komentar ini kemudian digunakan untuk menganalisis ulang, mendesain ulang, dan mengimplementasikan kembali prototipe kedua, yang menyediakan beberapa fitur lagi. Proses ini berlanjut dalam siklus sampai analis, pengguna, dan sponsor setuju bahwa prototipe menyediakan fungsionalitas yang cukup untuk diinstal dan digunakan dalam organisasi. Setelah prototipe (sekarang disebut "sistem") diinstal, penyempurnaan dilakukan hingga sistem ini diterima sebagai sistem baru (lihat Gambar 1-5).

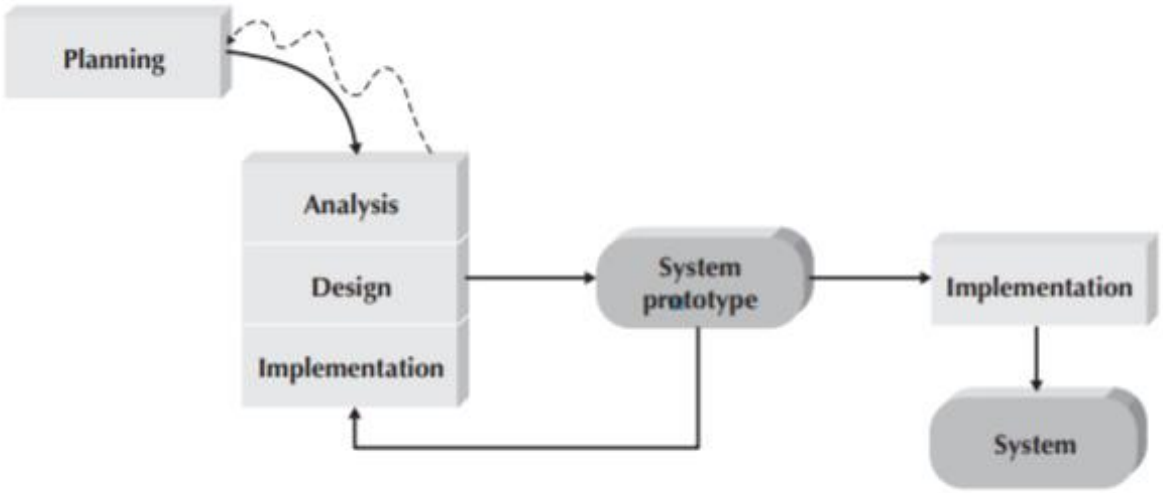


GAMBAR 1-4 Metodologi Berbasis Pengembangan Bertahap

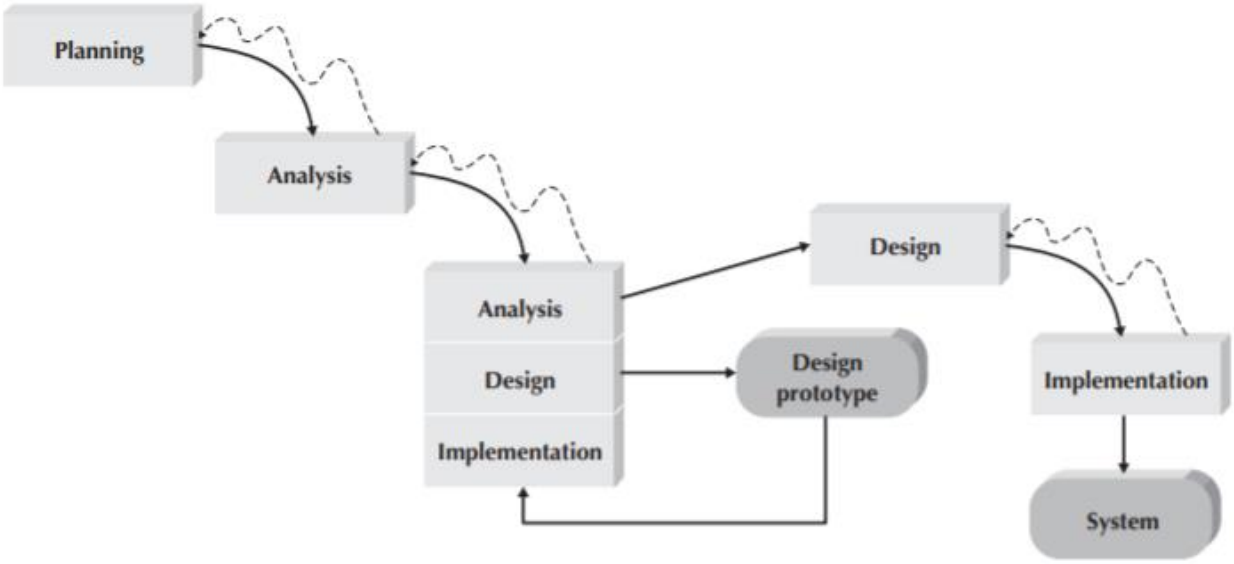
Keuntungan utama dari metodologi berbasis prototyping adalah sangat cepat menyediakan sistem yang dapat berinteraksi dengan pengguna, bahkan jika pada awalnya memang tidak siap untuk penggunaan organisasi secara luas. Pembuatan prototipe meyakinkan pengguna bahwa tim proyek sedang mengerjakan sistem (tidak ada jeda lama sehingga pengguna tetap dapat melihat sedikit kemajuan), dan pembuatan prototipe membantu menyempurnakan persyaratan nyata dengan lebih cepat.

Masalah utama dengan pembuatan prototipe adalah bahwa sistemnya yang bergerak cepat sehingga menantang analis sistem untuk melakukan analisis metodis secara hati-hati. Seringkali prototipe mengalami perubahan signifikan sehingga beberapa keputusan desain awal dapat menjadi keputusan yang buruk. Hal ini dapat menyebabkan masalah dalam pengembangan sistem yang kompleks karena isu-isu dan masalah mendasar tidak dikenali hingga masuk ke dalam proses pengembangan. Bayangkan membuat sebuah mobil dan terlambat menyadari dalam proses pembuatan prototipe bahwa Anda harus mengeluarkan seluruh mesin untuk mengganti oli (karena tidak ada yang berpikir tentang perlunya mengganti oli sampai kendaraan dikendarai hingga 10.000 mil).

Prototyping Sekali Pakai. Metodologi berbasis prototyping sekali pakai ini mirip dengan metodologi berbasis prototyping yang mencakup pengembangan prototipe; namun, prototipe sekali pakai dilakukan pada titik yang berbeda dalam SDLC. Prototipe ini digunakan dengan tujuan yang sangat berbeda dari yang telah dibahas sebelumnya, dan prototipe ini memiliki penampilan yang sangat berbeda (lihat Gambar 1-6).



GAMBAR 1-5 Metodologi Berbasis Prototyping



GAMBAR 1-6 Metodologi Berbasis Prototyping Sekali Pakai

Metodologi berbasis prototipe sekali pakai memiliki fase analisis yang relatif menyeluruh yang digunakan untuk mengumpulkan informasi dan mengembangkan ide untuk konsep sistem. Namun, pengguna mungkin tidak sepenuhnya memahami banyak fitur yang disediakan oleh metodologi ini, dan mungkin ada masalah teknis yang menantang untuk dipecahkan. Masing-masing masalah ini diperiksa dengan menganalisis, merancang, dan membangun prototipe desain. Prototipe desain bukanlah sistem kerja; ini adalah produk yang mewakili bagian dari sistem yang membutuhkan penyempurnaan tambahan, dan hanya berisi detail yang cukup untuk memungkinkan pengguna memahami masalah yang sedang dipertimbangkan. Misalnya, pengguna tidak sepenuhnya jelas tentang bagaimana cara kerja sistem entri pesanan. Dalam hal ini, serangkaian layar tiruan muncul seperti sebuah sistem, tetapi sebenarnya tidak melakukan apa-apa. Atau misalkan tim proyek perlu mengembangkan program grafis canggih di Java. Tim itu dapat menulis sebagian program dengan data pura-pura untuk memastikan bahwa mereka dapat melakukan program lengkap secara sukses.

Sistem yang dikembangkan menggunakan metodologi jenis ini bergantung pada beberapa prototipe desain selama fase analisis dan desain. Masing-masing prototipe

digunakan untuk meminimalkan risiko yang terkait dengan sistem dengan mengonfirmasi bahwa isu-isu penting dipahami terlebih dulu sebelum sistem yang sebenarnya dibangun. Setelah masalah diselesaikan, proyek kemudian pindah ke desain dan implementasi. Pada titik ini, prototipe desain dibuang, yang merupakan perbedaan penting antara metodologi ini dan metodologi prototyping, dalam hal prototipe perkembangan menjadi sistem final.

Metodologi berbasis prototipe sekali pakai menyeimbangkan manfaat antara analisis yang dipikirkan dengan matang dan fase desain dengan keuntungan menggunakan prototipe untuk memperbaiki masalah utama sebelum sistem dibangun. Diperlukan waktu lebih lama untuk mencapai sistem akhir dibandingkan dengan metodologi berbasis prototipe, tetapi jenis metodologi ini biasanya menghasilkan sistem yang lebih stabil dan reliabel.

Pengembangan Agile[5]

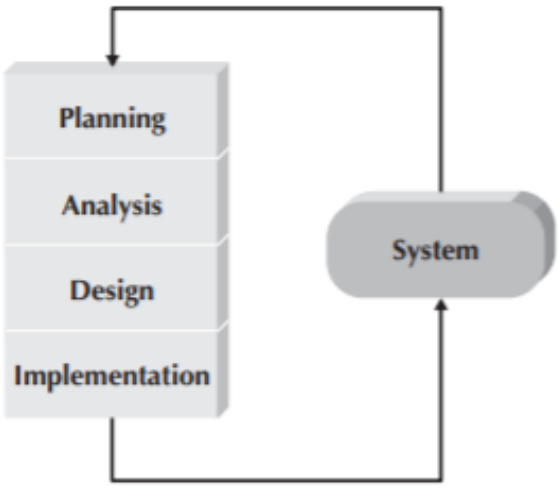
Kategori ketiga dari metodologi pengembangan sistem masih digunakan hingga saat ini, yaitu pengembangan agile. Semua metodologi pengembangan agile didasarkan pada manifesto agile dan dua belas prinsip. Penekanan manifestonya yaitu untuk memfokuskan pengembang pada kondisi kerja developer, perangkat lunak yang berfungsi, pelanggan, dan menangani perubahan persyaratan alih-alih fokus pada proses pengembangan sistem terperinci, alat, dokumentasi lengkap, kontrak hukum, dan detail rencana. Metodologi yang berpusat pada pemrograman ini memiliki sedikit aturan dan praktik, yang semuanya cukup mudah diikuti. Metodologi ini biasanya hanya didasarkan pada dua belas prinsip perangkat lunak agile. Prinsip-prinsip tersebut antara lain sebagai berikut:

- Perangkat lunak dikirimkan lebih awal dan melalui proses pengembangan secara terus menerus untuk memuaskan pelanggan.
- Perubahan persyaratan dipatuhi terlepas dari kapan itu terjadi dalam proses pengembangan.
- Perangkat lunak yang berfungsi dikirim terus menerus ke pelanggan.
- Pelanggan dan pengembang bekerja sama untuk memecahkan masalah bisnis.
- Individu yang termotivasi dapat membuat solusi; berikan mereka alat dan lingkungan yang mereka butuhkan, dan percayalah pada mereka.
- Komunikasi tatap muka dalam tim pengembangan adalah metode pengumpulan persyaratan yang paling efisien dan efektif.
- Ukuran utama kemajuan adalah bekerja untuk menyelesaikan perangkat lunak.
- Baik pelanggan maupun pengembang harus bekerja dengan kecepatan yang berkelanjutan. Artinya, tingkat pekerjaan dapat dipertahankan secara tak terbatas tanpa adanya kelelahan pekerja.
- Kelincahan ditingkatkan melalui fokus pada keunggulan teknis dan desain yang baik.
- Kesederhanaan, menghindari pekerjaan yang tidak perlu, sangat penting.
- Tim yang mengatur diri sendiri dapat mengembangkan arsitektur, persyaratan, dan desain terbaik.
- Tim pengembangan secara teratur memikirkan mengenai bagaimana meningkatkan proses pengembangan mereka.

Berdasarkan prinsip-prinsip ini, metodologi agile fokus dalam merampingkan proses pengembangan sistem dengan menghilangkan banyak pemodelan dan dokumentasi pengeluaran tambahan serta waktu yang dihabiskan untuk tugas-tugas tersebut. Sebaliknya,

[5] Tiga sumber informasi yang baik tentang pengembangan agile dan sistem berorientasi objek adalah: S. W. Ambler, Agile Modeling: Effective Practices for Extreme Programming and the Unified Process (New York: Wiley, 2002); C. Larman, Agile & Iterative Development: A Manager's Guide (Boston: Addison-Wesley, 2004); R. C. Martin, Agile Software Development: Principles, Patterns, and Practices (Upper Saddle River, NJ: Prentice Hall, 2003).

proyek sebisa mungkin sederhana, pengembangan aplikasi berulang.[6] Semua metodologi pengembangan agile mengikuti siklus yang sederhana melalui fase tradisional dari suatu proses pengembangan sistem (lihat Gambar 1-7). Hampir semua metodologi agile digunakan bersama dengan teknologi berorientasi objek.



GAMBAR 1-7 Metodologi Pengembangan Agile Tipikal

Namun, metodologi agile tetap memiliki cela. Salah satu cela utama berkaitan dengan lingkungan bisnis saat ini, di mana banyak dari pengembangan sistem informasi saat ini yang lepas pantai, outsourcing, dan/atau subkontrak. Mengingat metodologi pengembangan agile membutuhkan co-lokasi tim pengembangan, ini tampaknya menjadi asumsi yang sangat tidak realistis. Cela utama kedua adalah pengembangan agile tidak dikelola dengan hati-hati, dan berdasarkan definisinya memang tidak sehingga proses pengembangan dapat berubah menjadi pendekatan prototipe yang mana lingkungan akan berubah menjadi "programmer liar" di mana programmer mencoba untuk meretas solusi bersama. Cela utama ketiga, berdasarkan kurangnya dokumentasi aktual yang dibuat selama pengembangan perangkat lunak, hal ini dapat menimbulkan masalah terhadap kemampuan audit sistem yang dibuat. Tanpa dokumentasi yang memadai, baik sistem maupun proses pengembangan sistem, tidak dapat dijamin. Cela utama keempat didasarkan pada apakah pendekatan agile dapat memberikan sistem mission-critical yang besar.

Bahkan dengan cela ini, mengingat potensi pendekatan agile untuk mengatasi backlog aplikasi dan untuk memberikan solusi tepat waktu dalam banyak masalah bisnis, pendekatan agile harus dipertimbangkan dalam beberapa kondisi. Selain itu, banyak teknik yang didorong dengan memperhatikan tujuan yang mendasari manifesto agile dan dua belas prinsip agile sangat berguna dalam pengembangan sistem berorientasi objek. Dua contoh metodologi pengembangan agile yang lebih populer adalah pemrograman ekstrem (XP) dan Scrum.

Extreme Programming.[7] Extreme programming (XP) didasarkan pada empat nilai inti: komunikasi, kesederhanaan, umpan balik, dan nyali. Keempat nilai ini memberikan landasan yang digunakan pengembang XP untuk membuat semua jenis sistem. Pertama, pengembang harus memberikan umpan balik yang cepat kepada pengguna secara terus menerus. Kedua,

[6] Lihat Agile Alliance, www.agilealliance.com

[7] Untuk informasi lebih lanjut, lihat K. Beck, eXtreme Programming Explained: Embrace Change (Reading, MA: AddisonWesley, 2000); C. Larman, Agile & Iterative Development: A Manager's Guide (Boston: Addison-Wesley, 2004); M. Lippert, S. Roock, and H. Wolf, eXtreme Programming in Action: Practical Experiences from Real World Projects (New York: Wiley, 2002); www.extremeprogramming.org.

XP mengharuskan pengembang untuk mengikuti prinsip KISS.[8] Ketiga, pengembang harus membuat perubahan bertahap untuk mengembangkan sistem, dan sistem itu tidak hanya harus menerima perubahan, sistem itu juga harus merangkul perubahan. Keempat, pengembang harus memiliki mentalitas yang mengutamakan kualitas. XP juga mendukung anggota tim dalam mengembangkan keterampilan mereka sendiri. Tiga prinsip utama yang digunakan XP untuk membuat sistem yang sukses adalah pengujian berkelanjutan, pengkodean sederhana yang dilakukan pengembang, dan interaksi yang erat dengan pengguna untuk membangun sistem dengan cepat.

Pengujian dan praktik pengkodean yang efisien adalah inti dari XP. Kode diuji setiap hari dan ditempatkan ke dalam lingkungan pengujian integratif. Jika ada bug, kode dicadangkan hingga benar-benar tidak ada kesalahan.

Proyek XP dimulai dengan cerita pengguna yang menjelaskan apa yang perlu dilakukan oleh sistem. Kemudian, progammer membuat kode dalam modul kecil yang sederhana dan melakukan pengujian untuk memenuhi persyaratan tersebut. Pengguna harus bersedia untuk menjelaskan pertanyaan dan masalah yang muncul. Standar dibutuhkan untuk meminimalkan kebingungan, jadi tim XP menggunakan serangkaian nama, deskripsi, dan praktik pengkodean yang umum. Proyek XP memberikan hasil lebih cepat daripada pendekatan RAD, dan proyek ini jarang terjebak dalam proses pengumpulan persyaratan untuk sistem.

Penganut XP mengklaim banyak kelebihan jika pengembangan perangkat lunak dikerjakan menggunakan XP. Programmer bekerja sama dengan semua pemegang kepentingan, dan komunikasi antara mereka dapat ditingkatkan. Pengujian terus-menerus dari sistem yang berkembang sangat dianjurkan. Sistem ini dikembangkan secara evolusioner dan inkremental, yang memungkinkan persyaratan sistem berkembang saat para pemegang kepentingan memahami potensi yang dimiliki teknologi dalam memberikan solusi untuk permasalahan mereka. Estimasi dalam XP ditentukan oleh tugas dan dilakukan oleh programmer yang akan mengimplementasikan solusi untuk tugas yang sedang mereka kerjakan. Karena semua pemrograman dilakukan secara berpasangan, tanggung jawab bersama untuk setiap komponen perangkat lunak merupakan tanggung jawab antara para programmer. Pada akhirnya, kualitas produk akhir meningkat di setiap iterasi.

Untuk proyek kecil dengan tim yang termotivasi, kohesif, stabil, dan berpengalaman, XP dapat dimanfaatkan dengan baik. Namun, jika proyeknya tidak kecil atau tim tidak kompak,[9] keberhasilan upaya pengembangan XP sangat kecil. Hal ini dapat memberi keraguan mengenai gagasan untuk membawa kontraktor luar ke dalam lingkungan tim yang menggunakan XP.[10] Peluang orang luar bersaing dengan orang dalam mungkin terlalu optimis. XP harus benar-benar disiplin, jika tidak, proyek tidak akan fokus dan kacau. XP direkomendasikan hanya untuk kelompok kecil developer—tidak lebih dari sepuluh developer—dan tidak disarankan untuk aplikasi besar yang sangat penting. Karena kurangnya analisis dan dokumentasi desain, hanya ada dokumentasi kode yang terkait dengan XP, sehingga memelihara sistem besar yang dibangun dengan XP sangat tidak mungkin. Dan karena sistem informasi bisnis mission-critical cenderung tersedia untuk waktu yang lama,

[8] Jadikan sederhana, bodoh.

[9] Tim kompak adalah tim yang memiliki turnover rendah, identitas yang kuat, elit, perasaan bahwa mereka bersama-sama memiliki produk yang dikembangkan, dan kenikmatan dalam bekerja sama. Untuk informasi lebih lanjut mengenai tim kompak, lihat T. DeMarco dan T. Lister, Peopleware: Productive Projects and Teams (New York: Dorset/House, 1987).

[10] Karena kecenderungan outsourcing lepas pantai, hal ini merupakan kendala utama bagi XP. Untuk informasi lebih lanjut tentang outsourcing lepas pantai, lihat P. Thibodeau, "ITAA Panel Debates Outsourcing Pro, Cons," Computerworld Morning Update (September 25, 2003); S.W. Ambler, "Chicken Little Was Right," Software Development (Oktober 2003).

kegunaan XP sebagai metodologi pengembangan sistem informasi bisnis dapat diragukan. Pada akhirnya, metodologi ini membutuhkan banyak masukan pengguna di tempat, sesuatu yang tidak dapat dilakukan oleh banyak unit bisnis.[11] Namun, beberapa teknik yang terkait dengan XP berguna dalam pengembangan sistem berorientasi objek. Misalnya, cerita pengguna, pemrograman berpasangan, dan pengujian berkelanjutan adalah alat yang sangat berharga yang dapat dimanfaatkan oleh pengembangan sistem berorientasi objek.

Scrum.[12] Scrum adalah istilah yang dikenal oleh para penggemar rugby. Dalam rugby, scrum digunakan untuk memulai kembali permainan. Singkatnya, pencipta metode Scrum percaya bahwa tidak peduli berapa banyak yang Anda rencanakan, begitu perangkat lunak mulai dikembangkan, kekacauan akan muncul dan rencana tersebut keluar dari jalur.[13] Hal terbaik yang dapat Anda lakukan adalah bereaksi ke mana bola rugby terlempar. Anda kemudian berlari dengan bola sampai scrum berikutnya. Dalam kasus metodologi Scrum, sprint dilakukan selama tiga puluh hari kerja. Di akhir sprint, sistem dikirimkan ke pelanggan.

Dari semua pendekatan pengembangan sistem di permukaan, Scrum adalah yang paling kacau. Untuk mengendalikan beberapa kekacauan bawaan, pengembangan Scrum berfokus pada beberapa praktik utama. Tim diatur sendiri dan diarahkan sendiri. Tidak seperti pendekatan lain, tim Scrum tidak memiliki pemimpin tim yang ditunjuk. Sebaliknya, tim mengatur diri mereka sendiri secara simbiosis dan menetapkan tujuan mereka sendiri untuk setiap sprint (iterasi). Setelah sprint dimulai, tim Scrum tidak mempertimbangkan persyaratan tambahan apa pun. Setiap persyaratan baru yang ditemukan, akan ditempatkan pada tumpukan persyaratan yang masih perlu ditangani. Di awal setiap hari kerja, rapat Scrum diadakan. Di akhir setiap sprint, tim mendemonstrasikan perangkat lunak kepada klien. Mengacu pada hasil sprint terakhir, rencana baru dimulai untuk sprint berikutnya.

Pertemuan Scrum adalah salah satu aspek yang paling menarik dari proses pengembangan Scrum. Anggota tim menghadiri pertemuan, tetapi siapa pun dapat hadir, tidak terbatas anggota saja. Namun, dengan sedikit pengecualian, hanya anggota tim yang boleh berbicara. Satu pengecualian yang menonjol adalah manajemen memberikan umpan balik tentang relevansi bisnis dari pekerjaan yang dilakukan oleh tim tertentu. Dalam pertemuan ini, semua anggota tim berdiri dalam lingkaran dan melaporkan apa yang mereka capai pada hari sebelumnya, menyatakan apa rencana mereka hari ini, dan menjelaskan apa pun yang menjadi kendala di hari sebelumnya. Agar memungkinkan kemajuan berkelanjutan, setiap blok yang diidentifikasi ditangani dalam waktu satu jam. Dari sudut pandang Scrum, lebih baik membuat keputusan "buruk" mengenai blok pada saat ini dalam pengembangan daripada tidak membuat keputusan sama sekali. Karena pertemuan berlangsung setiap hari, keputusan buruk dapat dengan mudah dibatalkan. Larman [14] menyarankan bahwa setiap anggota tim harus melaporkan persyaratan tambahan apa pun yang ditemukan selama sprint dan apa pun yang dipelajari anggota tim yang dapat berguna untuk anggota tim lainnya.

---

[11] Pengamatan tentang kegunaan XP sebagai pendekatan pengembangan didasarkan pada percakapan dengan Brian Henderson-Sellers.

[12] Untuk informasi lebih lanjut, lihat C. Larman, Agile & Iterative Development: A Manager's Guide (Boston: Addison-Wesley, 2004); K. Schwaber dan M. Beedle, Agile Software Development with Scrum (Upper Saddle River, NJ: Prentice Hall, 2001); R. Wysocki, Effective Project Management: Traditional, Agile, Extreme, 5th Ed. (Indianapolis, IN: Wiley Publishing, 2009).

[13] Pengembang scrum bukanlah yang pertama kali mempertanyakan penggunaan rencana. Salah satu pepatah favorit Presiden Eisenhower adalah, "Dalam mempersiapkan pertempuran, saya selalu menemukan bahwa rencana tidak berguna, tetapi perencanaan sangat diperlukan." M. Dobson, Streetwise Project Management: How to Manage People, Processes, and Time to Achieve the Results You Need (Avon, MA: F+W Publications, 2003), hal. 43

[14] C. Larman, Agile & Iterative Development: A Manager's Guide (Boston: Addison-Wesley, 2004).

Salah satu kritik utama terhadap Scrum, seperti halnya semua metodologi agile, adalah apakah Scrum dapat ditingkatkan untuk mengembangkan sistem yang sangat besar, yaitu sistem mission-critical. Ukuran tim Scrum umumnya tidak lebih dari tujuh anggota. Satu-satunya prinsip pengorganisasian yang diajukan oleh pengikut Scrum untuk mengatasi cela ini adalah mengatur scrum dari suatu scrum. Setiap tim bertemu setiap hari, dan setelah pertemuan tim berlangsung, perwakilan (bukan pemimpin) dari setiap tim menghadiri pertemuan scrum-of-scrums. Hal ini berlanjut sampai kemajuan seluruh sistem ditentukan. Tergantung pada jumlah tim yang terlibat, pendekatan ini untuk mengelola proyek besar diragukan. Namun, seperti di XP dan pendekatan pengembangan agile lainnya, banyak ide dan teknik yang terkait dengan pengembangan Scrum yang berguna dalam pengembangan sistem berorientasi objek, seperti fokus pertemuan Scrum, pendekatan evolusioner dan inkremental untuk mengidentifikasi persyaratan, dan pendekatan inkremental dan iteratif untuk pengembangan sistem.

Memilih Metodologi Pengembangan yang Tepat

Karena banyaknya metodologi, tantangan pertama yang dihadapi analis adalah memilih metodologi yang akan digunakan. Memilih metodologi tidaklah sederhana, karena tidak ada metodologi yang terbaik dibanding lainnya. (Jika ya, kita akan menggunakannya di mana-mana!) Banyak organisasi memiliki standar dan kebijakan dalam memilih metodologi. Anda akan menemukan bahwa organisasi ada yang memiliki satu metodologi yang "disetujui", ada pula organisasi yang memiliki beberapa opsi metodologi, dan bahkan ada yang tidak memiliki kebijakan formal sama sekali.

Kemampuan untuk Mengembangkan Sistem	Metodologi Terstruktur		Metodologi RAD			Metodologi Agile	
	Waterfall	Paralel	Fase	Prototyping	Prototyping Sekali Pakai	XP	SCRUM
Dengan Persyaratan Pengguna yang Tidak Jelas	Buruk	Buruk	Baik	Sangat Baik	Sangat Baik	Sangat Baik	Sangat Baik
Dengan Teknologi yang Tidak Familiar	Buruk	Buruk	Baik	Buruk	Sangat Baik	Baik	Baik
Kompleks	Baik	Baik	Baik	Buruk	Sangat Baik	Baik	Baik
Reliabel	Baik	Baik	Baik	Buruk	Sangat Baik	Sangat Baik	Sangat Baik
Dengan Jadwal yang Singkat	Buruk	Baik	Sangat Baik	Sangat Baik	Baik	Sangat Baik	Sangat Baik
Dengan Visibilitas Jadwal	Buruk	Buruk	Sangat Baik	Sangat Baik	Baik	Sangat Baik	Sangat Baik

GAMBAR 1-8 Kriteria Pemilihan Metodologi

Gambar 1-8 merangkum beberapa kriteria penting untuk memilih metodologi. Satu hal penting yang tidak dibahas dalam gambar ini adalah tingkat pengalaman tim analis. Beberapa metodologi berbasis RAD memerlukan penggunaan alat dan teknik baru yang membutuhkan waktu belajar yang signifikan. Seringkali alat dan teknik ini meningkatkan kompleksitas proyek dan membutuhkan waktu ekstra untuk belajar. Namun, begitu hal itu diadopsi dan tim telah berpengalaman, alat dan teknik dapat secara signifikan meningkatkan kecepatan metodologi dalam membentuk sistem akhir.

Kejelasan Persyaratan Pengguna. Ketika persyaratan pengguna untuk suatu sistem tidak jelas, tentu sulit untuk memahaminya dengan membicarakannya dan menjelaskannya melalui laporan tertulis. Pengguna biasanya perlu berinteraksi dengan teknologi untuk benar-benar memahami apa yang dapat dilakukan sistem baru tsb dan mengetahui cara terbaik untuk menerapkannya pada kebutuhan mereka. Metodologi RAD dan agile biasanya lebih tepat jika persyaratan penggunanya tidak jelas.

Keakraban dengan Teknologi. Ketika sistem akan menggunakan teknologi baru yang tidak begitu familiar bagi para analis dan programmer, penerapan lebih awal teknologi baru dalam metodologi akan meningkatkan peluang keberhasilan. Jika sistem dirancang tanpa pengetahuan tentang teknologi dasar, risiko bisa meningkat karena alat mungkin tidak mampu melakukan apa yang dibutuhkan. Metodologi berbasis prototipe sekali pakai sangat tepat jika pengguna kurang familiar dengan teknologi karena metodologi tsb secara eksplisit mendorong pengembang untuk mengembangkan prototipe desain untuk area berisiko tinggi. Metodologi berbasis pengembangan bertahap menciptakan peluang untuk menyelidiki teknologi secara mendalam sebelum desain diselesaikan. Selain itu, karena sifat metodologi agile yang berpusat pada pemrograman, baik XP dan Scrum bisa digunakan. Meskipun Anda mungkin berpikir bahwa metodologi berbasis prototyping juga tepat, tetapi metodologi ini jauh lebih sedikit karena prototipe awal yang dibangun biasanya hanya menyentuh permukaannya saja dari teknologi baru tsb. Biasanya, setelah beberapa prototipe dibuat dan setelah beberapa bulan, pengembang akan menemukan kelemahan atau masalah dalam teknologi baru tsb.

Kompleksitas Sistem. Sistem yang kompleks memerlukan analisis dan desain yang cermat dan terperinci. Metodologi berbasis prototyping sekali pakai sangat cocok untuk analisis dan desain rinci seperti itu, tetapi metodologi berbasis prototyping tidak. Metodologi berbasis desain terstruktur tradisional dapat menangani sistem yang kompleks, tetapi tidak memiliki kemampuan untuk memasukkan sistem atau prototipe ke tangan pengguna sejak awal, beberapa masalah utama mungkin dapat terabaikan. Meskipun metodologi berbasis pengembangan bertahap memungkinkan pengguna untuk berinteraksi dengan sistem di awal proses, kami telah mengamati bahwa tim proyek yang mengikuti metodologi ini cenderung mencurahkan lebih sedikit perhatian pada analisis domain dari keseluruhan masalah dibanding jika mereka menggunakan metodologi lain. Pada akhirnya, metodologi agile bisa disebut sebagai campuran tas dalam hal kompleksitas sistem. Jika sistem berukuran besar, kinerja metodologi agile akan memburuk. Namun, jika sistemnya berukuran kecil atau sedang, maka pendekatan agile akan sangat baik. Kami menilai metodologi tsb baik dalam kriteria ini.

Reliabilitas Sistem. Reliabilitas sistem biasanya merupakan faktor penting dalam pengembangan sistem; lagipula, siapa yang menginginkan sistem yang tidak reliabel? Namun, reliabilitas hanyalah salah satu faktor di antara beberapa faktor. Untuk beberapa aplikasi, reliabilitas sangat penting (misalnya, peralatan medis, sistem kontrol rudal), sedangkan untuk aplikasi lain (misalnya, game, video internet) juga penting namun tidak begitu. Karena metodologi prototyping sekali pakai menggabungkan analisis rinci dan fase desain dengan kemampuan tim proyek untuk menguji banyak pendekatan yang berbeda melalui prototipe desain sebelum menyelesaikan desain, metodologi ini cocok ketika reliabilitas sistem sangat diprioritaskan. Metodologi prototipe umumnya bukan pilihan yang baik ketika reliabilitas

sangat diprioritaskan karena tidak memiliki analisis yang cermat dan fase desain yang penting untuk sistem yang dapat diandalkan. Namun, karena sangat difokuskan pada pengujian, identifikasi kebutuhan evolusioner dan inkremental, dan pengembangan iteratif dan inkremental, metode agile mungkin merupakan pendekatan yang terbaik.

Jadwal yang Singkat. Metodologi berbasis RAD dan agile adalah pilihan yang sangat baik jika waktu yang disediakan pendek karena keduanya paling memungkinkan tim proyek untuk menyesuaikan fungsionalitas dalam sistem didasarkan pada tanggal pengiriman tertentu, dan jika jadwal proyek mulai molor, hal itu bisa disesuaikan lagi dengan cara menghapus fungsionalitas dari versi atau prototipe yang sedang dikembangkan. Metodologi waterfall adalah pilihan terburuk ketika waktu sangat diprioritaskan karena metodologi ini tidak memungkinkan perubahan jadwal dengan mudah.

Visibilitas Jadwal. Salah satu tantangan terbesar dalam pengembangan sistem adalah menentukan apakah proyek berjalan sesuai jadwal. Hal ini utamanya berlaku untuk metodologi desain terstruktur karena desain dan implementasi dilakukan pada akhir proyek. Metodologi berbasis RAD mengerjakan banyak keputusan desain penting di awal dalam proyek untuk membantu manajer proyek mengenali dan mengatasi faktor risiko dan menjaga agar ekspektasi tetap terkendali. Namun, mengingat pertemuan progress harian yang terkait dengan pendekatan Agile, visibilitas jadwal selalu menjadi prioritas utama.

1.5 PERAN DAN KETERAMPILAN ANALIS SISTEM KHUSUS

Dari berbagai fase dan langkah yang dilakukan selama SDLC, jelas bahwa tim proyek membutuhkan berbagai keterampilan. Anggota proyek adalah agen perubahan yang mengidentifikasi cara untuk meningkatkan organisasi, membangun sistem informasi untuk mendukungnya, dan melatih dan memotivasi orang lain untuk menggunakan sistem. Memahami apa yang harus diubah dan bagaimana mengubahnya—dan meyakinkan orang lain tentang perlunya perubahan—membutuhkan berbagai keterampilan. Keterampilan ini dapat dipecah menjadi enam kategori utama: teknis, bisnis, analitis, interpersonal, manajemen, dan etika.

Analis harus memiliki keterampilan teknis untuk memahami lingkungan teknis organisasi yang ada, teknologi yang akan membentuk sistem baru, dan cara keduanya dapat masuk ke dalam solusi teknis terintegrasi. Keterampilan bisnis diperlukan untuk memahami bagaimana TI dapat diterapkan pada situasi bisnis dan untuk memastikan bahwa TI memberikan nilai bisnis yang nyata. Analis adalah pemecah masalah yang berkelanjutan di tingkat proyek dan organisasi, dan mereka harus menguji kemampuan analitis mereka secara teratur.

Analis sering kali perlu berkomunikasi secara efektif satu lawan satu dengan pengguna dan manajer bisnis (yang sering kali memiliki sedikit pengalaman dengan teknologi) dan dengan programmer (yang memiliki keahlian teknis lebih banyak daripada analis). Mereka harus mampu memberikan presentasi kepada kelompok besar dan kecil dan menulis laporan. Mereka tidak hanya perlu memiliki kemampuan interpersonal yang kuat, tetapi mereka juga perlu mengelola orang-orang yang bekerja dengan mereka dan mereka perlu mengelola tekanan dan risiko dalam situasi yang tidak jelas.

Pada akhirnya, analis harus adil, jujur, dan etis terhadap anggota tim proyek lainnya, manajer, dan pengguna sistem. Analis sering berurusan dengan informasi rahasia atau informasi yang, jika dibagikan dengan orang lain, dapat menyebabkan kerugian (misalnya, perbedaan pendapat di antara karyawan); penting untuk menjaga kepercayaan diri dan kepercayaan terhadap semua orang.

Selain enam keahlian umum ini, analis memerlukan banyak keterampilan khusus yang terkait dengan peran yang mereka lakukan pada sebuah proyek. Pada awal-awal pengembangan sistem, kebanyakan organisasi mengharapkan satu orang analis dengan semua keterampilan khusus yang diperlukan untuk melakukan proyek pengembangan sistem. Beberapa organisasi kecil masih mengandalkan satu orang untuk melakukan banyak peran, tetapi karena organisasi dan teknologi menjadi lebih kompleks, sebagian besar organisasi besar sekarang membangun tim proyek yang berisi beberapa individu dengan tanggung jawab yang jelas. Organisasi yang berbeda membagi peran secara berbeda. Sebagian besar tim SI mencakup banyak individu lain, seperti programmer, yang benar-benar menulis program untuk membangun sistem, dan penulis teknis, yang menyiapkan layar bantuan dan dokumentasi lainnya (misalnya, manual pengguna dan manual sistem).

Analis Bisnis

Seorang analis bisnis fokus pada isu-isu bisnis seputar sistem. Isu-isu ini termasuk mengidentifikasi nilai bisnis yang akan dibuat oleh sistem, mengembangkan ide dan saran mengenai bagaimana proses bisnis dapat ditingkatkan, dan merancang proses dan kebijakan baru bersama dengan analis sistem. Individu ini mungkin memiliki pengalaman bisnis dan beberapa jenis pelatihan profesional. Dia mewakili kepentingan sponsor proyek dan pengguna utama sistem. Seorang analis bisnis membantu dalam fase perencanaan dan desain tetapi paling besar dalam fase analisis.

Analis Sistem

Seorang analis sistem fokus pada isu-isu SI seputar sistem. Orang ini mengembangkan ide dan saran mengenai bagaimana teknologi informasi dapat meningkatkan proses bisnis, mendesain proses bisnis baru dengan bantuan analis bisnis, mendesain sistem informasi baru, dan memastikan bahwa semua standar SI dikelola. Seorang analis sistem kemungkinan memiliki pelatihan dan pengalaman yang signifikan dalam analisis dan desain, pemrograman, dan bahkan dalam bidang bisnis. Dia mewakili kepentingan departemen SI dan bekerja secara intensif melalui proyek tetapi mungkin tidak banyak dalam fase implementasi.

Analis Infrastruktur

Analis infrastruktur berfokus pada masalah teknis seputar bagaimana sistem akan berinteraksi dengan infrastruktur teknis organisasi (misalnya, perangkat keras, perangkat lunak, jaringan, dan database). Tugas analis infrastruktur termasuk memastikan bahwa sistem informasi baru sesuai dengan standar organisasi dan mengidentifikasi perubahan infrastruktur yang diperlukan untuk mendukung sistem. Analis ini mungkin memiliki pelatihan dan pengalaman yang signifikan dalam jaringan, administrasi database, dan berbagai produk perangkat keras dan perangkat lunak. Dia mewakili kepentingan organisasi dan kelompok SI yang pada akhirnya harus mengoperasikan dan mendukung sistem baru setelah dipasang. Seorang analis infrastruktur bekerja di seluruh proyek tetapi mungkin tidak begitu sering selama fase perencanaan dan analisis.

Analis Manajemen Perubahan

Seorang analis manajemen perubahan fokus pada orang-orang dan masalah manajemen seputar instalasi sistem. Peran orang ini juga termasuk memastikan bahwa dokumentasi dan dukungan yang memadai tersedia bagi pengguna, memberikan pelatihan kepada pengguna tentang sistem baru, dan mengembangkan strategi untuk mengatasi penolakan terhadap perubahan yang dibuat. Analis ini harus memiliki pelatihan dan pengalaman yang signifikan mengenai sifat organisasi pada umumnya dan manajemen perubahan pada khususnya. Dia mewakili kepentingan sponsor proyek dan pengguna. Seorang analis manajemen perubahan

bekerja paling aktif selama fase implementasi tetapi mulai meletakkan dasar kerja untuk perubahan selama fase analisis dan desain.

Manajer Proyek

Seorang manajer proyek bertanggung jawab untuk memastikan bahwa proyek selesai tepat waktu dan sesuai anggaran dan bahwa sistem memberikan semua kegunaan yang diminta oleh sponsor proyek. Peran manajer proyek termasuk mengelola anggota tim, mengembangkan rencana proyek, menugaskan sumber daya, dan menjadi kontak utama ketika orang-orang di luar tim memiliki pertanyaan mengenai proyek. Orang ini kemungkinan besar memiliki pengalaman yang signifikan dalam manajemen proyek dan mungkin telah bekerja selama bertahun-tahun sebagai analis sistem sebelumnya. Dia mewakili kepentingan departemen SI dan sponsor proyek. Manajer proyek bekerja secara intens pada semua fase proyek.

1.6 KARAKTERISTIK DASAR SISTEM BERORIENTASI OBJEK

Sistem berorientasi objek fokus dalam menangkap struktur dan perilaku sistem informasi dalam modul kecil yang mencakup data dan proses. Modul kecil ini dikenal sebagai objek. Pada bagian ini, kami menjelaskan karakteristik dasar sistem berorientasi objek, yang meliputi kelas, objek, metode, pesan, enkapsulasi, penyembunyian informasi, pewarisan, polimorfisme, dan pengikatan dinamis.[15]

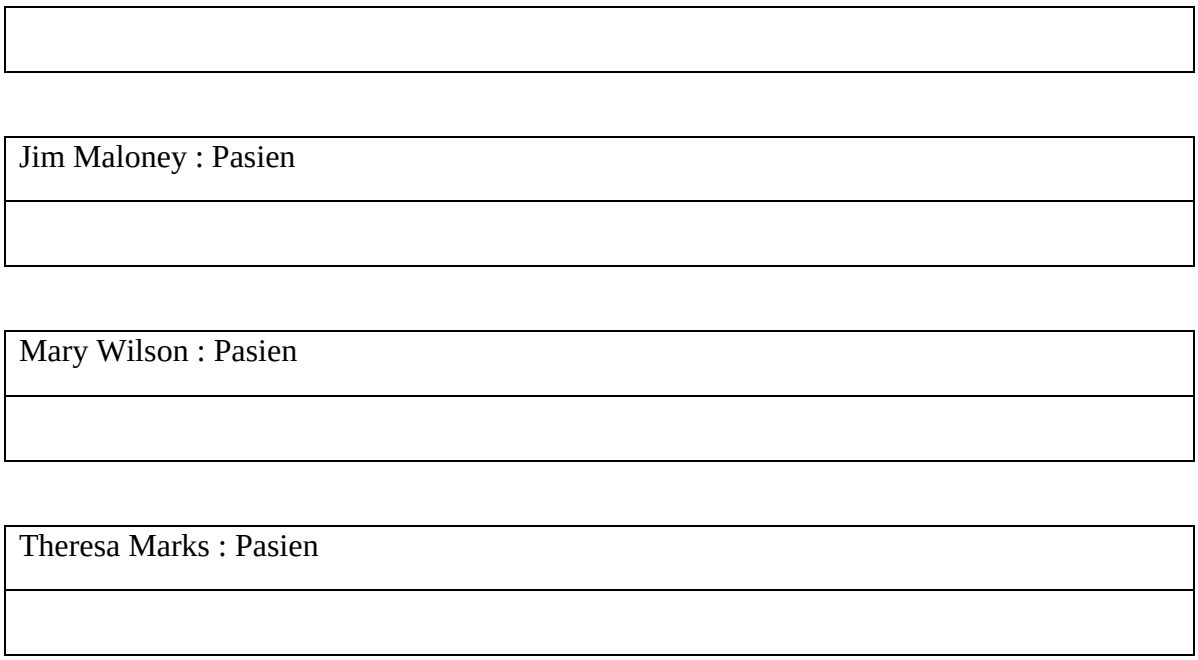
Kelas dan Objek

Kelas adalah template umum yang kita gunakan untuk mendefinisikan dan membuat contoh tertentu atau objek. Setiap objek diasosiasikan dengan sebuah kelas. Misalnya, semua objek yang memiliki informasi mengenai pasien dapat masuk ke dalam kelas yang disebut Pasien, karena ada atribut (misalnya, nama, alamat, tanggal lahir, telepon, dan operator asuransi) dan metode (misalnya, membuat janji, menghitung kunjungan terakhir, mengubah status, dan menyediakan riwayat medis) yang semuanya dibagikan oleh pasien (lihat Gambar 1-9).

Objek adalah instantiasi dari kelas. Dengan kata lain, objek adalah orang, tempat, atau benda yang ingin kita peroleh informasinya. Jika kita sedang membuat sebuah sistem janju untuk pertemuan kantor dokter, kelasnya adalah Dokter, Pasien, dan Janji Pertemuan. Pasien seperti Jim Maloney, Mary Wilson, dan Theresa Marks, dianggap sebagai contoh, atau objek dari kelas pasien (lihat Gambar 1-9).

Pasien
- nama - alamat - nomor telepon - asuransi
+ membuat janji() + menghitung kunjungan terakhir() + mengubah status() + menyediakan rekam medis() + create()

[15] Di bab berikutnya, kami meninjau karakteristik dasar sistem berorientasi objek secara lebih rinci.



GAMBAR 1-9 Kelas dan Objek

Setiap objek memiliki atribut yang menggambarkan informasi mengenai objek tersebut, seperti nama pasien, tanggal lahir, alamat, dan nomor telepon. Atribut juga digunakan untuk mewakili hubungan antar objek; misalnya, mungkin ada atribut departemen di objek karyawan dengan nilai objek departemen yang memiliki informasi di departemen mana objek karyawan bekerja. Keadaan suatu objek ditentukan oleh nilai atributnya dan hubungannya dengan objek lain pada waktu tertentu. Misalnya, seorang pasien mungkin memiliki keadaan baru atau keadaan saat ini atau keadaan sebelumnya.

Setiap objek juga memiliki perilaku. Perilaku ini menentukan apa yang dapat dilakukan objek. Misalnya, objek janji pertemuan mungkin dapat menjadwalkan janji baru, menghapus janji, dan membuat janji pertemuan berikutnya. Dalam pemrograman berorientasi objek, perilaku diimplementasikan sebagai metode (lihat bagian berikutnya).

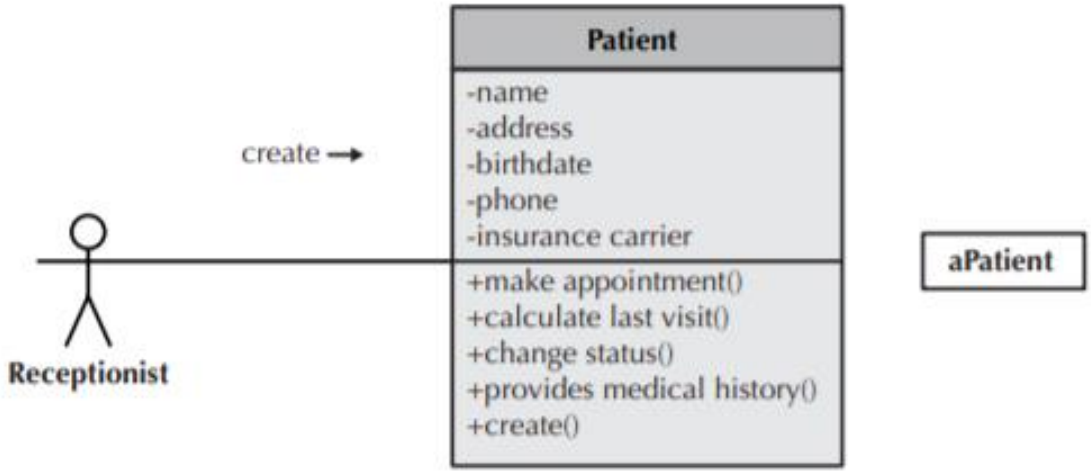
Salah satu aspek yang lebih membingungkan dari pengembangan sistem berorientasi objek adalah fakta bahwa di sebagian besar bahasa pemrograman berorientasi objek, baik kelas maupun contoh dari kelas dapat memiliki atribut dan metode. Atribut dan metode kelas cenderung digunakan untuk memodelkan atribut (atau metode) yang menangani masalah yang terkait dengan semua contoh kelas. Misalnya, untuk membuat objek pasien baru, pesan dikirim ke kelas Pasien untuk membuat contoh baru dari dirinya sendiri. Namun, dalam buku ini, kami memfokuskan terutama pada atribut dan metode objek dan bukan kelas.

Metode dan Pesan

Metode berguna untuk mengimplementasikan perilaku objek. Metode sebenarnya tidak lebih dari sebuah tindakan yang dapat dilakukan oleh objek. Pesan adalah informasi yang dikirim ke objek untuk memicu metode. Pesan pada dasarnya adalah panggilan fungsi atau prosedur dari satu objek ke objek lain. Misalnya, jika pasiennya merupakan pasien baru di sebuah kantor dokter, resepsionis mengirim pesan create ke aplikasi. Kelas pasien menerima pesan create tsb dan mengeksekusi metode create() yang kemudian membuat objek baru: aPatient (lihat Gambar 1-10).

Enkapsulasi dan Penyembunyian Informasi

Ide enkapsulasi dan penyembunyian informasi saling terkait dalam sistem berorientasi objek. Namun, tidak satu pun dari istilah tersebut yang baru. Enkapsulasi hanyalah kombinasi dari proses dan data dalam satu kesatuan. Penyembunyian informasi pertama kali dikerjakan dalam pengembangan sistem terstruktur. Prinsip penyembunyian informasi adalah hanya informasi yang diperlukan untuk menggunakan modul perangkat lunak yang ditunjukkan kepada pengguna modul. Biasanya, hal ini menyiratkan bahwa informasi yang diterbitkan adalah informasi yang diperlukan untuk diteruskan ke modul dan informasi yang dikirim dari modul. Hal mengenai bagaimana modul mengimplementasikan fungsionalitas yang diperlukan tidak begitu penting. Kami benar-benar tidak peduli bagaimana objek menjalankan fungsinya, selama fungsi itu benar-benar terjadi. Dalam sistem berorientasi objek, menggabungkan enkapsulasi dengan prinsip penyembunyian informasi sama halnya dengan memperlakukan objek sebagai kotak hitam.



GAMBAR 1-10 Pesan dan Metode

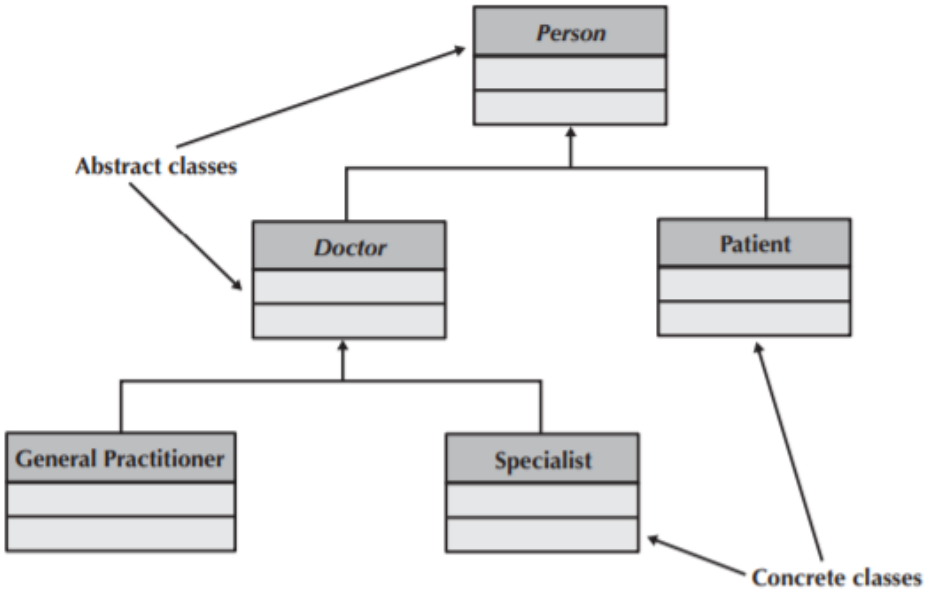
Fakta bahwa kita dapat menggunakan objek dengan memanggil metode adalah utama reusabilitas karena dapat melindungi kerja internal objek dari perubahan di sistem luar, dan menjaga sistem agar tidak terpengaruh saat perubahan dilakukan pada objek. Pada Gambar 1-10, perhatikan bagaimana pesan (create) dikirim ke objek, namun algoritma internal yang diperlukan untuk menanggapi pesan disembunyikan dari bagian lain dari sistem tsb. Satu-satunya informasi yang perlu diketahui objek adalah kumpulan operasi, atau metode, yang dapat dilakukan objek lain dan pesan apa yang perlu dikirim untuk memicunya.

Pewarisan

Pewarisan, sebagai suatu karakteristik pengembangan sistem informasi, diusulkan dalam pemodelan data pada akhir tahun 1970-an dan awal tahun 1980-an. Literatur pemodelan data menyarankan penggunaan pewarisan untuk mengidentifikasi kelas objek tingkat yang lebih tinggi atau yang lebih umum. Kumpulan atribut dan metode umum dapat diatur ke dalam superclass. Biasanya, kelas diatur dalam hierarki di mana superclass atau kelas umum berada di atas, dan subclass, atau kelas khusus, berada di bawah. Pada Gambar 1-11, Person adalah superclass dari kelas Doctor dan Patient. Doctor, pada gilirannya, adalah superclass untuk General Practitioner dan Specialist. Perhatikan bagaimana sebuah kelas (misalnya, Doctor) dapat berfungsi sebagai superclass dan subclass secara bersamaan. Hubungan antara kelas dan superclass-nya dikenal sebagai hubungan sejenis. Sebagai contoh pada Gambar 1-11, General Practitioner adalah sejenis Doctor, yang mana juga sejenis Person.

Subclass mewarisi atribut dan metode yang sesuai dari superclass di atasnya. Artinya, setiap subclass berisi atribut dan metode dari superclass induknya. Sebagai contoh, Gambar 1-11 menunjukkan bahwa Doctor dan Patient adalah subclass dari Person dan oleh karena itu mewarisi atribut dan metode dari kelas Person. Pewarisan membuatnya lebih mudah untuk mendefinisikan kelas. Alih-alih mengulang atribut dan metode di kelas Doctor dan Patient secara terpisah, atribut dan metode yang umum untuk keduanya ditempatkan di kelas Person dan diwarisi oleh kelas di bawahnya. Perhatikan bahwa hierarki pewarisan kelas objek jauh lebih efisien dibandingkan tanpa hierarki pewarisan (lihat Gambar 1-12).

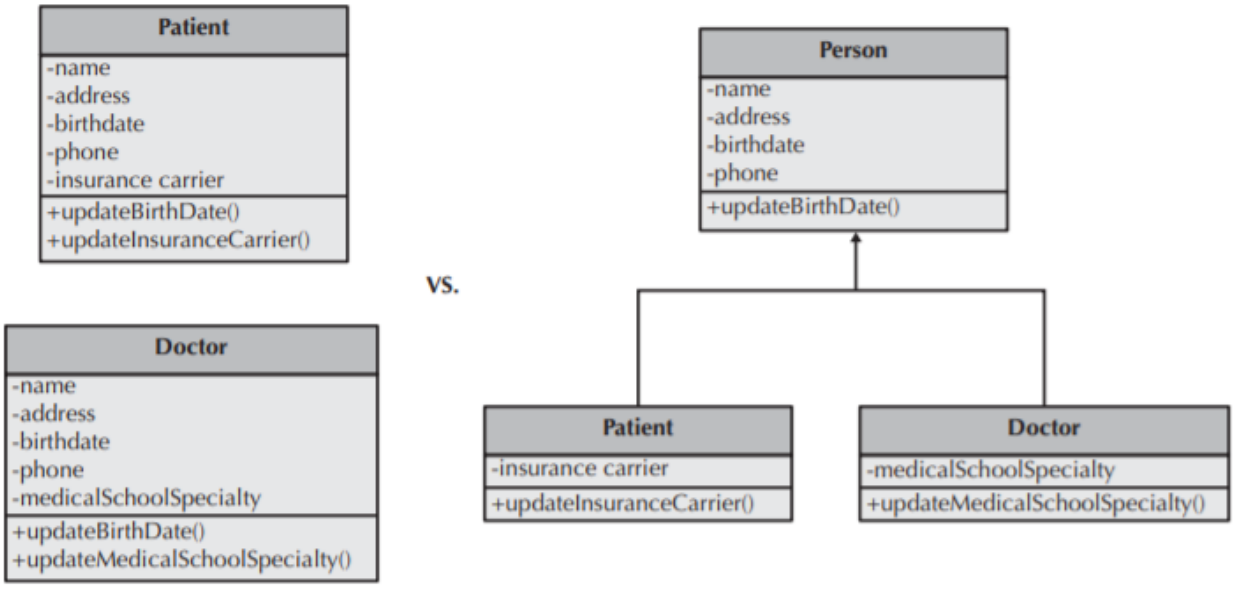
Sebagian besar kelas di seluruh hierarki mengarah ke contoh; setiap kelas yang memiliki contoh disebut kelas konkret. Misalnya, jika Mary Wilson dan Jim Maloney adalah turunan dari kelas Pasien, Pasien akan dianggap sebagai kelas konkret (lihat Gambar 1-9). Beberapa kelas tidak menghasilkan contoh karena kelas itu hanya digunakan sebagai template untuk kelas lain yang lebih spesifik (terutama kelas yang letaknya tinggi dalam hierarki). Kelas-kelas tersebut disebut sebagai kelas abstrak. Orang adalah contoh dari kelas abstrak. Alih-alih membuat objek dari Person, kami membuat contoh yang mewakili kelas Spesialis dan Pasien yang lebih spesifik, keduanya merupakan tipe Person (lihat Gambar 1-11).



GAMBAR 1-11 Hirarki Kelas dengan Kelas Abstrak dan Konkret

Polimorfisme dan Pengikatan Dinamis

Polimorfisme artinya pesan yang sama dapat diinterpretasikan secara berbeda oleh kelas objek yang berbeda. Misalnya, memasukkan pasien berarti berbeda dengan memasukkan janji. Oleh karena itu, informasi yang berbeda perlu dikumpulkan dan disimpan. Untungnya, kita tidak perlu memikirkan bagaimana hal itu terjadi ketika menggunakan objek. Kita cukup mengirim pesan ke suatu objek, dan objek itu akan bertanggung jawab untuk menafsirkan pesan dengan tepat. Misalnya, jika seorang seniman mengirim pesan Gambarlah diri Anda ke objek persegi, objek lingkaran, dan objek segitiga, hasilnya akan sangat berbeda, meskipun pesannya sama. Perhatikan pada Gambar 1-13 bagaimana setiap objek merespons dengan tepat (dan berbeda) meskipun pesannya identik.



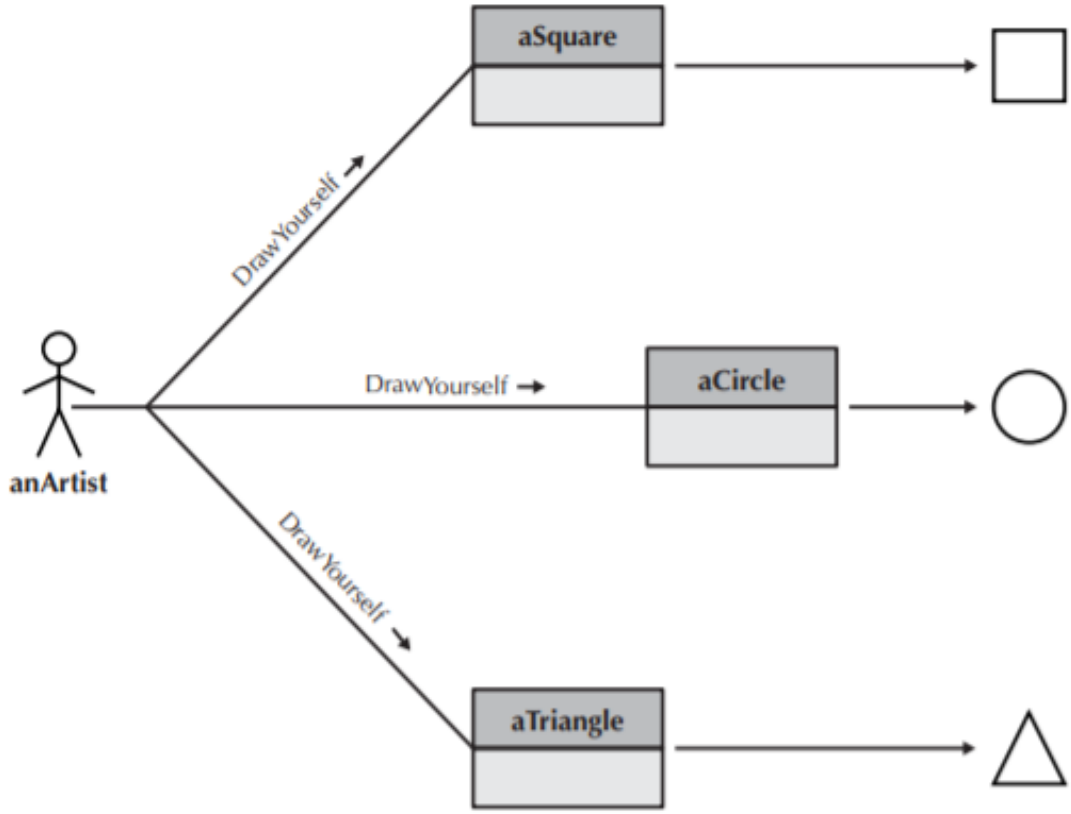
GAMBAR 1-12 Kelebihan Pewarisan?

Polimorfisme dapat dilakukan melalui pengikatan dinamis. Dinamis, atau yang awalnya disebut pengikatan adalah teknik menunda untuk mengetik objek hingga run-time. Metode spesifiknya tidak dipilih oleh sistem berorientasi objek hingga sistem berjalan. Hal ini berbeda dengan pengikatan statis. Dalam sistem yang terikat secara statis, jenis objek ditentukan pada saat waktu kompilasi. Oleh karena itu, developer harus memilih metode mana yang harus digunakan daripada harus membiarkan sistem melakukan hal itu. Inilah sebabnya mengapa sebagian besar bahasa pemrograman tradisional memiliki logika keputusan yang rumit yang didasarkan pada jenis objek yang berbeda dalam suatu sistem. Misalnya, dalam bahasa pemrograman tradisional, alih-alih mengirim pesan Gambarkan diri Anda ke berbagai jenis objek grafis pada Gambar 1-13, kita harus menulis logika keputusan menggunakan pernyataan kasus atau serangkaian if statement untuk menentukan jenis objek grafis yang ingin kita gambar, dan kita harus menamai setiap fungsi gambar secara berbeda (misalnya, menggambar persegi, menggambar lingkaran, atau menggambar segitiga). Hal ini jelas membuat sistem menjadi lebih rumit dan sulit untuk dipahami.

1.7 ANALISIS DAN DESAIN SISTEM BERORIENTASI OBYEK/OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN (OOSAD)

Pendekatan berorientasi objek untuk mengembangkan sistem informasi, secara teknis, dapat menggunakan salah satu metodologi tradisional. Namun, pendekatan berorientasi objek merupakan pendekatan yang paling berhubungan dengan pengembangan bertahap RAD atau metodologi agile. Perbedaan utama antara pendekatan tradisional seperti desain terstruktur dan pendekatan berorientasi objek adalah pada cara menguraikan masalah. Dalam pendekatan tradisional, proses dekomposisi masalahnya merupakan proses-sentris atau data-sentris. Namun, proses dan data sangat erat kaitannya sehingga sulit untuk memilih salah satunya sebagai fokus utama. Karena kurangnya kesesuaian dengan yang terjadi di dunia nyata, metodologi berorientasi objek versi baru muncul dengan menggunakan urutan fase SDLC berbasis RAD tetapi berusaha menyeimbangkan penekanan antara proses dan data dengan memfokuskan dekomposisi masalah pada objek yang berisi data sekaligus proses.

Menurut pencipta Unified Modeling Language (UML), Grady Booch, Ivar Jacobson, dan James Rumbaugh,[16] setiap pendekatan berorientasi objek modern yang berguna untuk mengembangkan sistem informasi harus berbasis Use-Case (use-case driven), arsitektur-sentris, dan berulang-bertahap.



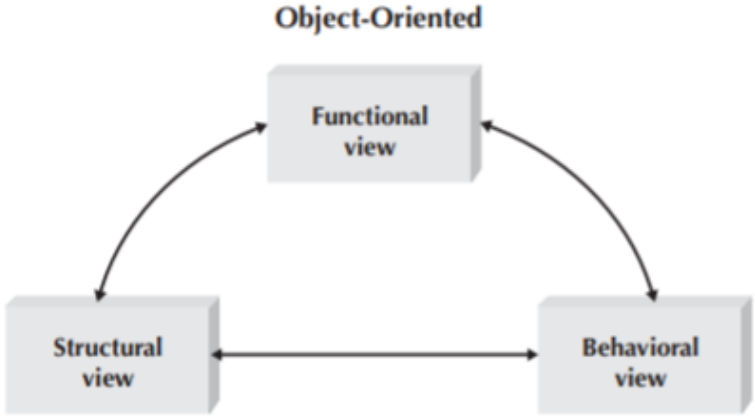
GAMBAR 1-13 Polimorfisme

Use-Case Driven

Use-case driven berarti Use-Casenya berupa alat pemodelan utama yang mendefinisikan perilaku sistem. Use-Case menjelaskan mengenai bagaimana pengguna berinteraksi dengan sistem untuk melakukan beberapa aktivitas, seperti menempatkan pesanan, membuat reservasi, atau mencari informasi. Use-Case digunakan untuk mengidentifikasi dan mengkomunikasikan persyaratan sistem kepada programmer yang harus menuliskan sistemnya. Use-Case secara inheren sederhana karena hanya fokus pada satu proses bisnis pada satu waktu. Sebaliknya, diagram model proses yang digunakan oleh metodologi terstruktur dan RAD tradisional jauh lebih kompleks karena memerlukan analis sistem dan pengguna untuk mengembangkan model sistem secara keseluruhan. Dengan metodologi tradisional, setiap sistem didekomposisi menjadi satu set subsistem yang pada gilirannya didekomposisi menjadi subsistem lebih lanjut dan seterusnya. Hal ini berlangsung sampai tidak ada lagi proses dekomposisi dan seringkali memerlukan puluhan halaman diagram yang saling terkait. Sebaliknya, Use-Case hanya fokus pada satu proses bisnis pada satu waktu, sehingga pengembangan modelnya jauh lebih sederhana.[17]

[16] Grady Booch, Ivar Jacobson, and James Rumbaugh, The Unified Modeling Language User Guide (Reading, MA: Addison-Wesley, 1999).

[17] Bagi Anda yang memiliki pengalaman dengan analisis dan desain terstruktur tradisional, ini adalah salah satu aspek yang paling tidak umum dari analisis dan desain berorientasi objek menggunakan UML. Tidak



GAMBAR 1-14 Pengembangan Berulang dan Bertahap

Arsitektur-Sentris

Setiap pendekatan modern untuk analisis dan desain sistem harus berupa arsitektur-sentris. Arsitektur-sentris artinya arsitektur perangkat lunak yang mendasari spesifikasi sistem yang sedang berkembang menentukan spesifikasi, konstruksi, dan dokumentasi sistem. Analisis dan pendekatan desain sistem berorientasi objek modern harus mendukung setidaknya tiga pandangan arsitektur sistem yang terpisah tetapi saling terkait: fungsional, statis, dan dinamis. Tampilan fungsional atau eksternal menggambarkan perilaku sistem dari perspektif pengguna. Pandangan struktural atau statis menggambarkan sistem dalam hal atribut, metode, kelas, dan hubungan. Tampilan perilaku atau dinamis menggambarkan perilaku sistem dalam hal pesan yang dikirimkan di antara objek dan perubahan status di dalam objek.

Berulang dan Bertahap

Pendekatan analisis dan desain dari sistem berorientasi objek modern menekankan pengembangan berulang dan bertahap yang mengalami pengujian dan penyempurnaan terus-menerus sepanjang proyek. Hal ini artinya analis sistem mengembangkan pemahaman mereka tentang masalah pengguna dengan membangun tiga tampilan arsitektur sedikit demi sedikit. Analis sistem melakukan ini bekerja sama dengan pengguna untuk membuat representasi fungsional dari sistem yang sedang dipelajari. Selanjutnya, analis mencoba membangun representasi struktural dari sistem yang berkembang. Dengan menggunakan representasi struktural dari sistem, analis mendistribusikan fungsionalitas sistem di atas struktur yang sedang berkembang untuk membuat representasi perilaku dari sistem tsb. Saat analis bekerja dengan pengguna dalam mengembangkan tiga tampilan arsitektur dari sistem yang berkembang, analis mengulangi setiap tampilan dan di antara tampilan tersebut. Artinya, ketika analis lebih memahami pandangan struktural dan perilaku, analis dapat mengungkap persyaratan yang hilang atau salah representasi dalam tampilan fungsional. Hal ini dapat menyebabkan perubahan melalui pandangan struktural dan perilaku. Ketiga tampilan arsitektur sistem saling terkait dan bergantung satu sama lain (lihat Gambar 1-14). Saat setiap tahapan dan pengulangan selesai, representasi yang lebih lengkap dari kebutuhan fungsional nyata pengguna diperoleh.

---

seperti pendekatan terstruktur, pendekatan berorientasi objek menekankan fokus hanya pada satu kasus penggunaan pada satu waktu dan mendistribusikan kasus penggunaan tunggal itu melalui sekumpulan objek yang berkomunikasi dan berkolaborasi.

Manfaat Analisis dan Desain Sistem Berorientasi Objek

Konsep dalam pendekatan berorientasi objek memungkinkan analis untuk memecah sistem yang kompleks menjadi modul yang lebih kecil dan lebih mudah dikelola, dikerjakan secara individual, dan dapat dengan mudah disatukan kembali untuk membentuk sistem informasi. Modularitas ini membuat pengembangan sistem jadi lebih mudah dipahami, lebih mudah dibagikan di antara anggota tim proyek, dan lebih mudah untuk dikomunikasikan dengan pengguna, yang dalam hal ini adalah orang yang menyediakan persyaratan dan yang juga mengonfirmasi seberapa baik sistem dalam memenuhi persyaratan selama proses pengembangan sistem. Dengan memodulasi pengembangan sistem, tim proyek sebenarnya sedang membuat bagian-bagian yang dapat digunakan kembali yang dapat dihubungkan ke sistem lain atau digunakan sebagai titik awal untuk proyek lain. Pada akhirnya, hal ini dapat menghemat waktu karena proyek baru tidak harus dimulai dari awal.

1.8 PROSES TERPADU (UNIFIED PROCESS)

Unified Process adalah metodologi spesifik yang memetakan kapan dan bagaimana menggunakan berbagai teknik Unified Modeling Language (UML) untuk analisis dan desain berorientasi objek. Kontributor utamanya adalah Grady Booch, Ivar Jacobsen, dan James Rumbaugh. Sementara UML memberikan dukungan struktural untuk mengembangkan struktur dan perilaku sistem informasi, Unified Process menyediakan dukungan perilaku. Unified Process adalah use-case driven, arsitektur-sentris, dan berulang-bertahap. Selanjutnya, Unified Process adalah proses pengembangan sistem dua dimensi yang digambarkan oleh serangkaian fase dan alur kerja. Fase-fase tersebut adalah awalan, elaborasi, konstruksi, dan transisi. Alur kerjanya meliputi pemodelan bisnis, persyaratan, analisis, desain, implementasi, pengujian, penyebaran, konfigurasi dan manajemen perubahan, manajemen proyek, dan lingkungan.[18] Gambar 1-15 menggambarkan Unified Process.

Fase

Fase Unified Process mendukung seorang analis dalam mengembangkan sistem informasi secara berulang dan bertahap. Fase tsb menggambarkan bagaimana sistem informasi berkembang sepanjang waktu. Tingkat aktivitas bisa bervariasi sepanjang alur kerja, bergantung di mana fase pengembangan sistem yang sedang berkembang saat itu. Kurva pada Gambar 1-15 menunjukkan alur kerja yang memperkirakan jumlah aktivitas selama fase tertentu. Misalnya, fase awal biasanya melibatkan pemodelan bisnis dan alur persyaratan, tetapi mengabaikan alur kerja pengujian dan penerapan. Setiap fase berisi satu set pengulangan, dan setiap pengulangan menggunakan berbagai alur kerja untuk membuat versi tambahan dari sistem yang sedang berkembang. Seiring sistem berkembang melalui fase itu, sistem tsb akan meningkat dan menjadi lebih lengkap. Setiap fase memiliki tujuan, fokus aktivitas (di atas alur kerja), dan kiriman tambahan. Masing-masing fase akan dijelaskan setelah ini.

Permulaan. Dalam banyak hal, fase permulaan sangat mirip dengan fase perencanaan dari suatu pendekatan SDLC tradisional. Pada fase ini, sebuah kasus bisnis dibuat untuk

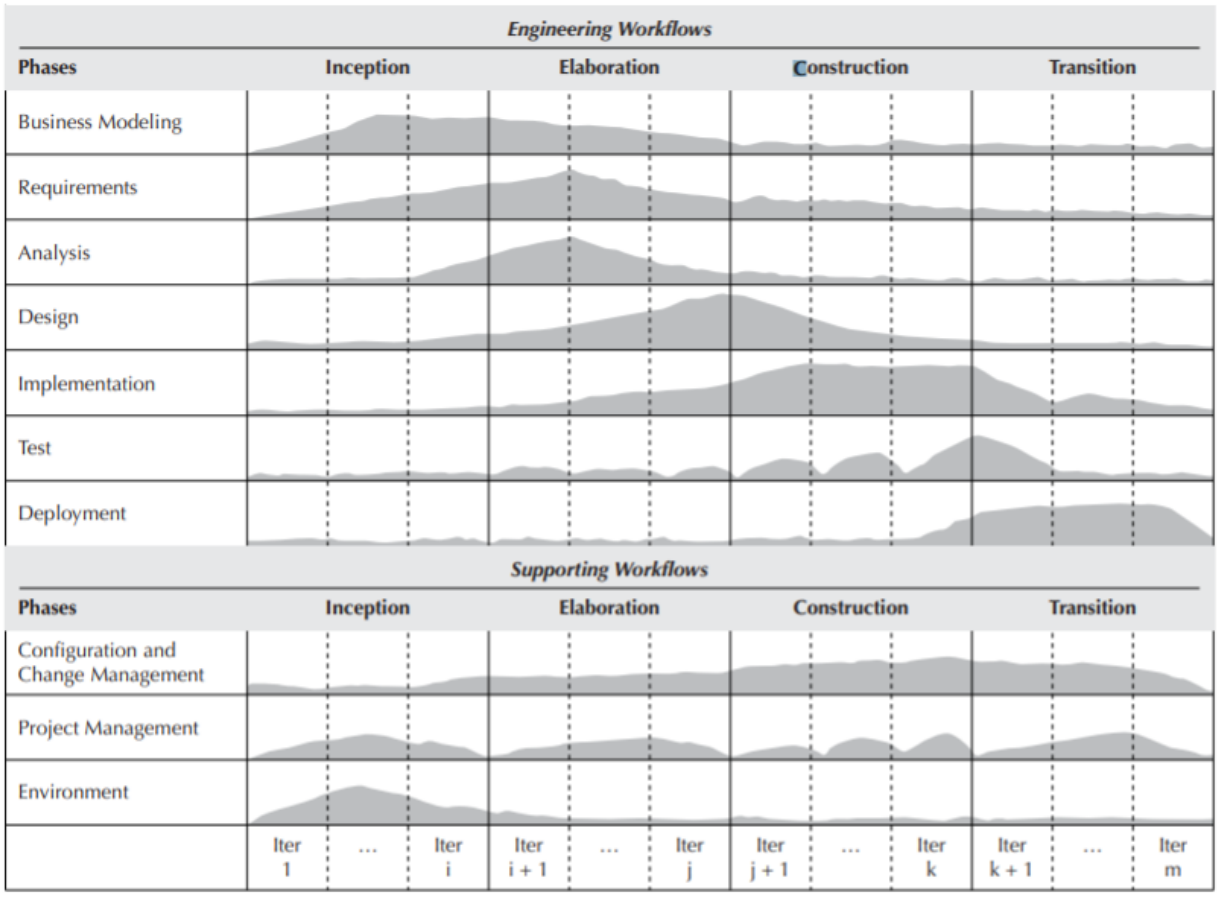
[18] Materi di bagian ini berdasarkan Khawar Zaman Ahmed dan Cary E. Umrysh, Developing Enterprise Java Applications with J2EE and UML (Boston, MA: Addison-Wesley, 2002); Jim Arlow dan Ila Neustadt, UML and The Unified Process: Practical Object-Oriented Analysis & Design (Boston, MA: Addison-Wesley, 2002); Peter Eeles, Kelli Houston, dan Wojtek Kozacynski, Building J2EE Applications with the Rational Unified Process (Boston, MA: Addison-Wesley, 2003); Ivar Jacobson, Grady Booch, dan James Rumbaugh, The Unified Software Development Process (Reading, MA: Addison-Wesley, 1999); Phillipe Krutchten, The Rational Unified Process: An Introduction, 2nd Ed. (Boston, MA: Addison-Wesley, 2000); "Rational Unified Process: Best Practices for Software Development Teams," Rational Software White Paper, TP026B, Rev 11/01.

sistem yang diusulkan. Hal ini termasuk analisis kelayakan yang harus bisa menjawab pertanyaan-pertanyaan seperti berikut:

Apakah kita memiliki kemampuan teknis untuk membangunnya (kelayakan teknis)?

Jika kita membangunnya, apakah dapat berdampak pada nilai bisnis (kelayakan ekonomi)?

Jika kita membangunnya, apakah akan digunakan oleh organisasi (kelayakan organisasi)?



GAMBAR 1-15 Unified Process (Proses Terpadu)

Untuk menjawab pertanyaan ini, tim pengembangan melakukan pekerjaan yang terkait dengan pemodelan bisnis, persyaratan, dan alur kerja analisis. Dalam beberapa kasus, tergantung pada kesulitan teknis yang mungkin dihadapi selama pengembangan sistem, prototipe sekali pakai dapat dikembangkan. Hal ini menyiratkan bahwa desain, implementasi, dan alur kerja pengujian juga dapat dilibatkan. Alur kerja yang mendukung manajemen proyek dan lingkungan sangat relevan dengan fase ini. Kiriman utama dari fase permulaan adalah dokumen visi yang menetapkan ruang lingkup proyek; mengidentifikasi persyaratan dan batasan utama; menyiapkan rencana proyek awal; dan menjelaskan kelayakan dan risiko yang terkait dengan proyek, adopsi lingkungan yang diperlukan untuk mengembangkan sistem, dan beberapa aspek dari kelas domain masalah yang diimplementasikan dan diuji.

Elaborasi. Ketika kita berpikir tentang analisis dan desain sistem berorientasi objek, aktivitas yang berkaitan dengan fase elaborasi dari Proses Terpadu adalah yang paling relevan. Alur kerja analisis dan desain adalah fokus utama dalam fase ini. Tahap elaborasi berlanjut dengan mengembangkan dokumen visi, termasuk menyelesaikan kasus bisnis, merevisi

penilaian risiko, dan menyelesaikan rencana proyek secara detail agar para pemegang kepentingan setuju untuk membangun sistem akhir yang sebenarnya. Hal ini berkaitan dengan pengumpulan persyaratan, membangun model struktural dan perilaku UML dari domain masalah, dan merinci bagaimana model domain masalah cocok dengan arsitektur sistem yang sedang berkembang. Developer terlibat dengan semuanya kecuali alur kerja teknik penyebaran dalam fase ini. Saat developer mengulangi alur kerja, pentingnya menangani konfigurasi dan manajemen perubahan menjadi lebih jelas. Selain itu, alat pengembangan yang diperoleh selama fase awal menjadi penting selama fase ini untuk keberhasilan proyek.[19] Hasil utama fase ini meliputi struktur UML dan diagram perilaku dan versi dasar dari sistem informasi yang sedang berkembang. Versi dasar berfungsi sebagai dasar untuk semua pengulangan selanjutnya. Dengan memberikan landasan yang kokoh pada titik ini, para developer memiliki dasar untuk menyelesaikan sistem dalam fase konstruksi dan transisi.

Konstruksi. Fase konstruksi sangat fokus pada pemrograman sistem informasi yang sedang berkembang. Fase ini berkaitan dengan alur kerja implementasi. Namun, alur kerja persyaratan dan alur kerja analisis dan desain juga terlibat dalam fase ini. Selama fase inilah persyaratan yang hilang diidentifikasi dan model analisis serta desain diselesaikan. Biasanya, ada pengulangan dari alur kerja selama fase ini, dan selama pengulangan terakhir, alur kerja penerapan ditingkatkan. Alur kerja konfigurasi dan manajemen perubahan, dengan aktivitas kontrol di versi tsb, menjadi sangat penting selama fase konstruksi. Terkadang, sebuah pengulangan harus diputar kembali. Tanpa kontrol versi yang baik, memutar kembali ke versi sebelumnya (implementasi bertahap) dari sistem hampir tidak mungkin dilakukan. Hasil utama dari fase ini adalah implementasi sistem yang dapat dirilis untuk pengujian beta dan uji penerimaan.

Transisi. Seperti fase konstruksi, fase transisi juga membahas aspek-aspek yang biasanya terkait dengan fase implementasi dari suatu pendekatan SDLC tradisional. Fokus utamanya pada alur kerja pengujian dan penerapan. Pada dasarnya, pemodelan bisnis, persyaratan, dan alur kerja analisis harus diselesaikan dalam pengulangan awal dari sistem informasi yang sedang berkembang. Selanjutnya, alur kerja pengujian akan dijalankan selama fase awal dari sistem tsb. Beberapa aktivitas seperti desain ulang dan pemrograman pada alur kerja desain dan implementasi mungkin diperlukan, bergantung pada hasil dari alur kerja pengujian, tetapi harus seminimal mungkin pada titik ini. Dari sudut pandang manajerial, lingkup-lingkup seperti manajemen proyek, konfigurasi dan manajemen perubahan, dan lingkungan juga terlibat. Beberapa aktivitas yang dilakukan adalah pengujian beta dan uji penerimaan, penyesuaian desain dan implementasi, pelatihan pengguna, dan peluncuran produk akhir ke platform produksi. Jelas, kiriman utamanya adalah sistem informasi aktual yang dapat dieksekusi. Kiriman lainnya termasuk manual pengguna, paket untuk mendukung pengguna, dan paket untuk meningkatkan sistem informasi di masa depan.

Alur Kerja

Alur kerja menggambarkan mengenai tugas atau aktivitas yang dilakukan oleh developer untuk mengembangkan sistem informasi dari waktu ke waktu. Alur kerja Proses Terpadu dikelompokkan ke dalam dua kategori besar, yaitu teknik dan pendukung

[19] Dengan UML yang terdiri dari lima belas teknik pembuatan diagram yang berbeda dan berkaitan, hal ini berguna untuk menjaga agar diagram tetap terkoordinasi dan versi yang berbeda dari sistem yang sedang berkembang secara sinkron biasanya di luar kemampuan developer. Alat ini biasanya mencakup manajemen proyek dan CASE. Kami menjelaskan penggunaan alat-alat ini di bab berikutnya.

Alur Kerja Teknik. Alur kerja teknik mencakup pemodelan bisnis, persyaratan, analisis, desain, implementasi, pengujian, dan alur kerja penerapan. Alur kerja teknik berhubungan dengan kegiatan yang menghasilkan produk teknis (yaitu, sistem informasi).

Alur Kerja Pemodelan Bisnis. Alur kerja pemodelan bisnis mengungkap mengenai masalah dan mengidentifikasi proyek potensial dalam organisasi pengguna. Alur kerja ini membantu manajemen dalam memahami ruang lingkup proyek yang berguna untuk meningkatkan efisiensi dan efektivitas organisasi pengguna. Tujuan utama dari pemodelan bisnis adalah untuk memastikan bahwa developer dan organisasi pengguna memahami di mana dan bagaimana sistem informasi yang akan dikembangkan cocok dengan proses bisnis organisasi pengguna. Alur kerja ini biasanya dijalankan selama fase awal untuk memastikan bahwa kita mengembangkan sistem informasi yang masuk akal secara bisnis. Kegiatan yang berlangsung pada alur kerja ini umumnya berkaitan dengan fase perencanaan SDLC tradisional; namun, pengumpulan persyaratan, dan teknik pemodelan Use-Case dan proses bisnis juga membantu kita dalam memahami situasi bisnis.

Alur Kerja Persyaratan. Dalam Proses Terpadu, alur kerja persyaratan mencakup persyaratan fungsional dan nonfungsional. Biasanya, persyaratan dikumpulkan dari pemegang kepentingan proyek, seperti pengguna akhir, manajer dalam organisasi pengguna akhir, dan bahkan pelanggan. Alur kerja persyaratan paling banyak digunakan selama fase awal dan fase elaborasi. Persyaratan yang telah diidentifikasi sangat membantu untuk mengembangkan dokumen visi dan Use-Case yang digunakan selama proses pengembangan. Persyaratan tambahan cenderung ditemukan selama proses pengembangan. Faktanya, hanya fase transisi yang cenderung memiliki sedikit persyaratan tambahan yang teridentifikasi.

Alur Kerja Analisis. Alur kerja analisis utamanya membahas mengenai pembuatan model analisis dari suatu domain masalah. Dalam Proses Terpadu, analis mulai merancang arsitektur yang terkait dengan domain masalah; dengan menggunakan UML, analis dapat membuat diagram struktural dan perilaku yang menggambarkan deskripsi kelas domain masalah dan interaksinya. Tujuan utama dari alur kerja analisis adalah untuk memastikan developer dan organisasi pengguna memahami masalah mendasar berserta domainnya tanpa menganalisis secara berlebihan. Jika mereka tidak hati-hati, analis justru dapat membuat kelumpuhan analisis, yang mana terjadi ketika proyek sangat macet terhadap analisis sehingga sistem tidak benar-benar dirancang atau diimplementasikan. Tujuan kedua dari alur kerja analisis adalah untuk mengidentifikasi kelas yang berguna yang dapat digunakan kembali untuk perpustakaan kelas. Dengan menggunakan kembali kelas yang telah ditentukan sebelumnya, analis dapat menghindari penemuan kembali metode saat membuat diagram struktural dan perilaku. Alur kerja analisis sebagian besar berhubungan dengan fase elaborasi, tetapi sebagaimana alur kerja persyaratan, ada kemungkinan bahwa analisis tambahan akan diperlukan selama proses pengembangan.

Alur Kerja Desain. Alur kerja desain mentransisikan model analisis ke dalam bentuk yang dapat digunakan untuk mengimplementasikan sistem, yaitu model desain. Sementara alur kerja analisis terkonsentrasi pada pemahaman domain masalah, alur kerja desain fokus pada pengembangan solusi yang akan dijalankan dalam lingkungan tertentu. Pada dasarnya, alur kerja desain hanya meningkatkan deskripsi sistem yang sedang berkembang dengan menambahkan kelas yang membahas lingkungan sistem ke model analisis tsb. Alur kerja desain merupakan aktivitas seperti desain kelas domain untuk masalah yang terperinci, optimalisasi sistem informasi yang sedang berkembang, desain database, desain antarmuka pengguna, dan desain arsitektur fisik. Alur kerja desain utamanya berhubungan dengan fase elaborasi dan konstruksi dari suatu Proses Terpadu.

Alur Kerja Implementasi. Tujuan utama dari alur kerja implementasi adalah untuk membuat solusi yang dapat dieksekusi berdasarkan model desain (yaitu, pemrograman). Hal ini tidak hanya penulisan kelas baru tetapi juga penggabungan kelas yang dapat digunakan kembali dari perpustakaan kelas yang dapat dieksekusi ke dalam solusi yang sedang berkembang. Seperti halnya aktivitas pemrograman, kelas baru dan interaksinya dengan kelas reusable harus diuji. Terakhir, dalam kasus seperti beberapa kelompok melakukan implementasi sistem informasi, pelaksana juga harus mengintegrasikan modul terpisah yang diuji masing-masing untuk membuat versi sistem yang dapat dieksekusi. Alur kerja implementasi utamanya berhubungan dengan fase elaborasi dan konstruksi.

Alur Kerja Pengujian. Tujuan utama dari alur kerja pengujian adalah untuk meningkatkan kualitas sistem yang sedang berkembang. Pengujian dilakukan dengan melewati pengujian unit sederhana yang terkait dengan alur kerja implementasi. Dalam hal ini, pengujian juga mencakup pengujian integrasi semua modul yang digunakan untuk mengimplementasikan sistem, pengujian penerimaan pengguna, dan pengujian alfa aktual perangkat lunak. Secara praktis, pengujian harus dilakukan selama pengembangan sistem; pengujian model analisis dan desain dilakukan selama fase elaborasi dan konstruksi, sedangkan pengujian implementasi dilakukan utamanya selama konstruksi dan fase transisi. Pada dasarnya, di akhir setiap pengulangan selama pengembangan sistem informasi, beberapa jenis pengujian harus dilakukan.

Alur Kerja Penerapan. Alur kerja penerapan sangat berhubungan dengan fase transisi dari suatu Proses Terpadu. Alur kerja penerapan mencakup aktivitas seperti pengemasan perangkat lunak, distribusi, pemasangan, dan pengujian beta. Ketika akan benar-benar menyebarkan sistem baru ke dalam organisasi pengguna, pengembang mungkin harus mengubah data yang sudah ada, menghubungkan perangkat lunak baru dengan perangkat lunak yang ada, dan melatih pengguna untuk menggunakan sistem baru.

Alur Kerja Pendukung. Alur kerja pendukung meliputi manajemen proyek, manajemen konfigurasi dan perubahan, dan alur kerja lingkungan. Alur kerja pendukung fokus pada aspek manajerial pengembangan sistem informasi.

Alur Kerja Manajemen Proyek. Sementara alur kerja lain yang terkait dengan Proses Terpadu secara teknis aktif dalam keempat fase, alur kerja manajemen proyek adalah satu-satunya alur kerja lintas fase yang sesungguhnya. Proses pengembangan mendukung pengembangan bertahap dan berulang, sehingga sistem informasi cenderung tumbuh atau berkembang dari waktu ke waktu. Di akhir setiap pengulangan, versi tambahan baru dari sistem siap untuk dikirim. Alur kerja manajemen proyek cukup penting karena adanya kompleksitas model pengembangan dua dimensi Proses Terpadu (alur kerja dan fase). Kegiatan alur kerja ini termasuk mengidentifikasi dan mengelola risiko, mengelola ruang lingkup, memperkirakan waktu untuk menyelesaikan setiap pengulangan dan keseluruhan proyek, memperkirakan biaya pengulangan individu dan keseluruhan proyek, dan melacak kemajuan menuju versi akhir dari sistem informasi yang sedang berkembang.

Alur Kerja Manajemen Konfigurasi dan Perubahan. Tujuan utama dari alur kerja manajemen konfigurasi dan perubahan adalah untuk melacak keadaan sistem yang sedang berkembang. Singkatnya, sistem informasi yang berkembang terdiri dari seperangkat artefak (misalnya, diagram, kode sumber, dan artefak lain yang bisa dieksekusi). Selama proses pengembangan, artefak ini dimodifikasi. Sejumlah besar pekerjaan — atau uang — terlibat dalam pengembangan artefak. Artefak itu sendiri harus ditangani seperti halnya aset yang mahal—kontrol akses harus diterapkan untuk melindungi artefak agar tidak dicuri atau dihancurkan. Selanjutnya, karena artefak dimodifikasi secara teratur, mekanisme kontrol yang baik harus dibuat. Pada akhirnya, informasi manajemen proyek yang baik juga harus diperoleh

(misalnya, penulis, waktu, dan lokasi setiap modifikasi). Alur kerja manajemen konfigurasi dan perubahan sebagian besar terkait dengan fase konstruksi dan transisi.

Alur Kerja Lingkungan. Selama pengembangan sistem informasi, tim developer perlu menggunakan alat dan proses yang berbeda. Alur kerja lingkungan menjawab kebutuhan ini. Misalnya, CASE tool yang dapat mendukung pengembangan sistem informasi berorientasi objek melalui UML mungkin diperlukan. Alat lain yang diperlukan antara lain lingkungan pemrograman, alat manajemen proyek, dan alat manajemen konfigurasi. Alur kerja lingkungan melibatkan keberadaan serta pemasangan alat-alat ini. Meskipun alur kerja ini aktif pada semua fase Proses Terpadu, namun alur ini harus terlibat lebih sering di fase awal.

Ekstensi untuk Proses Terpadu

Karena besarnya dan rumitnya Proses Terpadu, banyak penulis yang menunjukkan serangkaian kelemahan kritisnya. Pertama, Proses Terpadu tidak membahas masalah kepegawaian, penganggaran, atau manajemen kontrak. Kegiatan-kegiatan ini secara eksplisit tidak termasuk dalam Proses Terpadu. Kedua, Proses Terpadu tidak membahas masalah yang berkaitan dengan pemeliharaan, operasi, atau dukungan produk setelah dikirimkan. Jadi, ini bukanlah proses perangkat lunak yang lengkap; ini hanyalah proses pengembangan. Ketiga, Proses Terpadu tidak membahas masalah lintas atau antar proyek. Mempertimbangkan pentingnya penggunaan kembali proses ini dalam pengembangan sistem berorientasi objek, dan fakta bahwa di banyak organisasi karyawan bekerja di beberapa proyek yang berbeda pada saat yang sama, mengabaikan masalah antar proyek adalah suatu kelalaian besar.

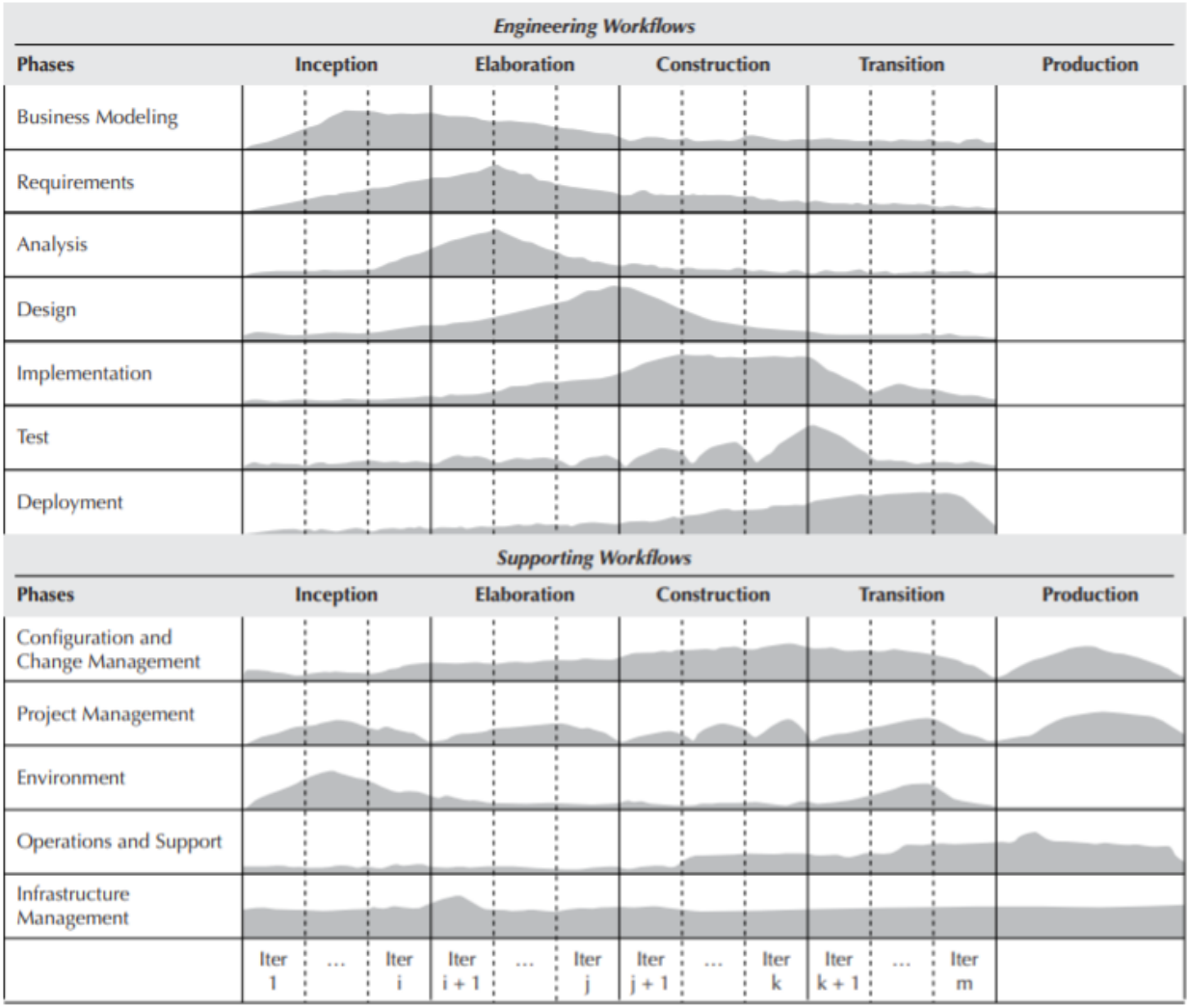
Untuk mengatasi masalah ini, Ambler dan Constantine menyarankan untuk menambahkan fase produksi dan dua alur kerja, yaitu alur kerja operasi dan dukungan dan alur kerja manajemen infrastruktur (lihat Gambar 1-16).[20] Selain alur kerja baru ini, pengujian, penyebaran, dan alur kerja lingkungan dimodifikasi, dan manajemen proyek serta alur kerja manajemen konfigurasi dan perubahan diperluas ke fase produksi. Ekstensi ini didasarkan pada proses perangkat lunak berorientasi objek alternatif, yaitu proses OPEN (Object-oriented Process, Environment, and Notation/Proses, Lingkungan, dan Notasi Berorientasi Objek) dan Proses Perangkat Lunak Berorientasi Objek.[21]

Fase Produksi. Fase produksi utamanya fokus pada isu-isu yang berkaitan dengan produk perangkat lunak setelah berhasil diluncurkan. Fase ini fokus pada isu-isu yang terkait dengan pembaruan, pemeliharaan, dan pengoperasian perangkat lunak. Tidak seperti fase sebelumnya, tidak ada pengulangan atau tambahan kiriman. Jika rilis baru dari perangkat lunak akan dikembangkan, maka pengembang harus mulai dari proses baru melalui empat fase pertama. Berdasarkan aktivitas yang dilakukan selama fase ini, tidak ada alur kerja teknis yang digunakan. Alur kerja pendukung yang aktif selama fase ini antara lain alur kerja

[20] S. W. Ambler dan L. L. Constantine, The Unified Process Inception Phase: Best Practices in Implementing the UP (Lawrence, KS: CMP Books, 2000); S. W. Ambler dan L. L. Constantine, The Unified Process Elaboration Phase: Best Practices in Implementing the UP (Lawrence, KS: CMP Books, 2000); S. W. Ambler dan L. L. Constantine, The Unified Process Construction Phase: Best Practices in Implementing the UP (Lawrence, KS: CMP Books, 2000); S. W. Ambler dan L. L. Constantine, The Unified Process Transition and Production Phases: Best Practices in Implementing the UP (Lawrence, KS: CMP Books, 2002).

[21] S. W. Ambler, Process Patterns—Building Large-Scale Systems Using Object Technology (Cambridge, UK: SIGS Books/Cambridge University Press, 1998); S. W. Ambler, More Process Patterns—Delivering Large-Scale Systems Using Object Technology (Cambridge, UK: SIGS Books/Cambridge University Press, 1999); I. Graham, B. Henderson-Sellers, dan H. Younessi, The OPEN Process Specification (Harlow, UK: Addison-Wesley, 1997); B. Henderson-Sellers dan B. Unhelkar, OPEN Modeling with UML (Harlow, UK: Addison-Wesley, 2000).

konfigurasi dan manajemen perubahan, alur kerja manajemen proyek, alur kerja operasi dan dukungan baru, dan alur kerja manajemen infrastruktur.



GAMBAR 1-16 Proses Terpadu yang Dtingkatkan

Alur Kerja Operasi dan Dukungan. Alur kerja operasi dan dukungan, seperti yang bisa Anda tebak, membahas masalah yang terkait dengan dukungan versi perangkat lunak saat ini dan pengoperasian perangkat lunak harian. Aktivitas antara lain membuat rencana untuk operasi dan dukungan produk perangkat lunak setelah digunakan, membuat pelatihan dan dokumentasi pengguna, menerapkan prosedur backup yang diperlukan, memantau dan mengoptimalkan kinerja perangkat lunak, dan melakukan pemeliharaan korektif pada perangkat lunak. Alur kerja ini dilakukan selama fase konstruksi; dan semakin meningkat sepanjang transisi hingga fase produksi. Alur kerja berhenti ketika versi perangkat lunak yang ada diganti dengan versi terbaru. Banyak developer salah sangka bahwa setelah perangkat lunak dikirimkan ke pelanggan, pekerjaan mereka selesai. Dalam kebanyakan kasus, dukungan produk perangkat lunak jauh lebih mahal dan memakan waktu daripada pengembangannya sendiri. Pada titik itu, pekerjaan developer mungkin baru saja dimulai.

Alur Kerja Manajemen Infrastruktur. Alur kerja manajemen infrastruktur bertujuan untuk mendukung pengembangan infrastruktur yang diperlukan untuk mengembangkan sistem berorientasi objek. Kegiatan seperti pengembangan dan modifikasi perpustakaan, standar, dan model perusahaan sangat penting. Jika pengembangan dan pemeliharaan model arsitektur suatu domain masalah melampaui lingkup proyek tunggal dan penggunaan kembali

dimungkinkan terjadi, alur kerja manajemen infrastruktur menjadi sangat penting. Serangkaian kegiatan lintas proyek yang tak kalah penting lainnya adalah peningkatan proses pengembangan perangkat lunak. Karena aktivitas pada alur kerja ini cenderung mempengaruhi banyak proyek, sementara Proses Terpadu hanya fokus pada proyek tertentu, Proses Terpadu cenderung mengabaikan aktivitas ini (artinya, aktivitas tersebut berada di luar cakupan dan tujuan Proses Terpadu).

Modifikasi dan Ekstensi Alur Kerja yang Sudah Ada. Selain alur kerja yang ditambahkan untuk mengatasi kekurangan yang terdapat dalam Proses Terpadu, alur kerja yang sudah ada harus dimodifikasi dan/atau diperluas ke fase produksi. Alur kerja ini mencakup pengujian, penerapan, lingkungan, manajemen proyek, dan alur kerja konfigurasi dan manajemen perubahan.

Alur Kerja Pengujian. Untuk sistem informasi berkualitas tinggi yang akan dikembangkan, pengujian harus dilakukan pada setiap kiriman, termasuk yang dibuat selama fase awal. Jika tidak, yang terbentuk adalah sistem yang kualitasnya tidak tinggi.

Alur Kerja Penyebaran. Sistem lama biasanya telah tersedia di sebagian besar perusahaan sekarang ini, dan sistem ini memiliki database terkait yang harus diubah agar bisa berinteraksi dengan sistem baru. Karena kerumitan penerapan sistem baru, konversi akan memerlukan perencanaan yang signifikan. Oleh karena itu, aktivitas pada alur kerja penyebaran perlu dimulai pada fase awal alih-alih menunggu hingga akhir fase konstruksi, seperti yang disarankan dalam Proses Terpadu.

Alur Kerja Lingkungan. Alur kerja lingkungan perlu dimodifikasi agar bisa memasukkan kegiatan yang terkait dengan pengaturan operasi dan lingkungan produksi. Pekerjaan yang dilakukan saat itu serupa dengan pekerjaan yang terkait dengan pengaturan lingkungan pengembangan di fase awal. Dalam hal ini, pekerjaan tambahan dilakukan selama fase transisi.

Alur Kerja Manajemen Proyek. Meskipun alur kerja manajemen proyek tidak termasuk mengatur staf proyek, mengelola kontrak antara pelanggan dan vendor, dan mengelola anggaran proyek, tetapi kegiatan ini sangat penting untuk keberhasilan setiap proyek pengembangan perangkat lunak. Kami menyarankan untuk memperluas manajemen proyek. Alur kerja ini juga harus dilakukan dalam fase produksi untuk mengatasi masalah seperti pelatihan, manajemen staf, dan manajemen hubungan klien.

Alur Kerja Konfigurasi dan Manajemen Perubahan. Alur kerja konfigurasi dan manajemen perubahan diperluas ke fase produksi baru. Kegiatan yang dilakukan selama fase produksi yaitu mengidentifikasi potensi perbaikan sistem operasional dan menilai dampak potensial dari perubahan yang diusulkan. Setelah developer mengidentifikasi perubahan ini dan memahami dampaknya, mereka dapat menjadwalkan perubahan yang akan dibuat dan diterapkan pada rilis mendatang.

Gambar 1-17 menunjukkan bab-bab yang mencantumkan fase dan alur kerja Proses Terpadu yang Ditingkatkan. Mengingat outsourcing lepas pantai dan otomatisasi teknologi informasi,[22] kita hanya akan fokus pada fase elaborasi dan pemodelan bisnis, persyaratan, analisis, desain, dan alur kerja manajemen proyek dari Proses Terpadu yang Ditingkatkan. Namun, seperti yang ditunjukkan Gambar 1-17, fase dan alur kerja lainnya dicantumkan. Di banyak lingkungan pengembangan sistem berorientasi objek saat ini, pembuatan kode

[22] Baca Thomas L. Friedman, The World Is Flat: A Brief History of the Twenty-First Century, Updated and Expanded Edition (New York: Farrar, Straus, and Giroux, 2006); Daniel H. Pink, A Whole New Mind: Why Right-Brainers Will Rule the Future (New York: Riverhead Books, 2006).

diperlukan. Oleh karena itu, dari perspektif bisnis, kami percaya bahwa kegiatan yang terkait dengan alur kerja ini merupakan kegiatan yang paling penting.

Fase Proses Terpadu yang Ditingkatkan	Bab
Permulaan	2-4
Elaborasi	3-11
Kostruksi	8, 12
Transisi	12-13
Produksi	13
Alur Kerja Teknis Proses Terpadu yang Ditingkatkan	Bab
Pemodelan Bisnis	2-5
Persyaratan	3-5, 10
Analisis	3-7
Desain	7-11
Implementasi	9, 12
Pengujian	4-7, 12
Penyebaran	13
Alur Kerja Pendukung Proses Terpadu yang Ditingkatkan	Bab
Manajemen Proyek	2, 13
Konfigurasi dan Manajemen Perubahan	13
Lingkungan	2
Operasi dan Dukungan	13
Manajemen Infrastruktur	2

GAMBAR 1-17 Proses Terpadu yang Ditingkatkan dan Organisasi Buku ini

1.9 UNIFIED MODELING LANGUAGE/BAHASA PEMODELAN TERPADU

Hingga tahun 1995, konsep objek memang populer tetapi diimplementasikan dengan cara berbeda-beda oleh developer yang berbeda. Setiap developer memiliki metodologi dan notasinya sendiri (misalnya, Booch, Coad, Moses, OMT, OOSE, SOMA).[23] Kemudian pada

[23] Baca Grady Booch, Object-Oriented Analysis and Design with Applications, 2nd Ed. (Redwood City, CA: Benjamin/ Cummings, 1994); Peter Coad dan Edward Yourdon, Object-Oriented Analysis, 2nd Ed. (Englewood Cliff s, NJ: Yourdon Press, 1991); Peter Coad dan Edward Yourdon, Object-Oriented Design (Englewood Cliffs, NJ: Yourdon Press, 1991); Brian Henderson-Sellers dan Julian Edwards, Book Two of Object-Oriented Knowledge: The Working Object (Sydney, Australia: Prentice Hall, 1994); James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, dan William Lorensen, Object-Oriented Modeling and

tahun 1995, Rational Software menyatukan tiga pemimpin industri untuk menciptakan pendekatan tunggal pada pengembangan sistem berorientasi objek. Grady Booch, Ivar Jacobson, dan James Rumbaugh bekerja dengan orang lain untuk membuat satu set standar teknik diagram yang dikenal sebagai Unified Modeling Language (UML). Tujuan UML adalah untuk menyediakan kosakata umum tentang istilah-istilah berorientasi objek dan teknik diagram untuk memodelkan proyek pengembangan sistem apa pun mulai dari tahap analisis hingga implementasi. Pada November tahun 1997, Object Management Group (OMG) secara resmi menerima UML sebagai standar untuk semua developer objek. Kemudian, pada tahun-tahun berikutnya, UML mengalami beberapa revisi kecil. Versi UML saat ini adalah Versi 2.5.

UML versi 2.5 ini mendefinisikan mengenai lima belas teknik diagram yang digunakan untuk memodelkan suatu sistem. Diagram dipecah menjadi dua kelompok besar: satu untuk memodelkan struktur sistem dan satu lagi untuk memodelkan perilaku. Diagram struktur menyediakan cara untuk menampilkan data dan hubungan statis dalam sistem informasi. Diagram struktur ini di antaranya adalah kelas, objek, paket, penyebaran, komponen, struktur komposit, dan diagram profil. Diagram perilaku membantu analis untuk menggambarkan hubungan dinamis antara contoh atau objek yang mewakili sistem informasi bisnis. Diagram itu juga memungkinkan pemodelan perilaku dinamis objek individu seiring waktu. Diagram perilaku mendukung analis dalam memodelkan persyaratan fungsional dari sistem informasi yang sedang berkembang. Diagram pemodelan perilaku mencakup aktivitas, urutan, komunikasi, tinjauan interaksi, pengaturan waktu, mesin status perilaku, mesin status protokol, dan diagram Use-Case.[24] Gambar 1-18 menunjukkan gambaran umum tentang diagram ini.

Nama Diagram	Digunakan untuk	Fase Utama
Diagram Struktur		
Kelas	Menggambarkan hubungan antar kelas yang dimodelkan dalam sistem	Analisis, Desain
Objek	Mengilustrasikan hubungan antara objek yang dimodelkan dalam sistem; digunakan ketika contoh aktual dari kelas dapat mengomunikasikan model dengan lebih baik	Analisis, Desain
Paket	Mengelompokkan elemen UML lainnya bersama-sama untuk membentuk konstruksi	Analisis, Desain, Implementasi

Design (Englewood Cliffs, NJ: Prentice Hall, 1991); Ivar Jacobson, Magnus Christerson, Patrik Jonsson, dan Gunnar Overgaard, Object-Oriented Software Engineering: A Use Case Approach (Wokingham, England: Addison-Wesley, 1992); Ian Graham, Migrating to Object Technology (Wokingham, England: Addison-Wesley, 1994).

[24] Materi yang disajikan dalam bagian ini didasarkan pada Unified Modeling Language: Superstructure Version 2.4, ptc/2010-11-14 (www.uml.org). Tambahan referensi, Michael Jesse Chonoles dan James A. Schardt, UML 2 for Dummies (Indianapolis, IN: Wiley, 2003); Hans-Erik Eriksson, Magnus Penker, Brian Lyons, dan David Fado, UML 2 Toolkit (Indianapolis, IN: Wiley, 2004); Kendall Scott, Fast Track UML 2.0 (Berkeley, CA: Apress, 2004). Untuk melihat penjelasan lengkap diagram, silakan buka www.uml.org.

	pada tingkat yang lebih tinggi	
Penyebaran	Menunjukkan arsitektur fisik sistem; juga dapat digunakan untuk menunjukkan komponen perangkat lunak yang digunakan ke arsitektur fisik	Desaik Fisik, Implementasi
Komponen	Mengilustrasikan hubungan fisik antara komponen-komponen perangkat lunak	Desaik Fisik, Implementasi
Desain Struktur Komposit	Menggambarkan struktur internal suatu kelas, yaitu hubungan antara bagian-bagian dari suatu kelas	Analisis, Desain
Profil	Digunakan untuk mengembangkan ekstensi ke UML itu sendiri	Tidak Ada
Diagram Perilaku		
Kegiatan	Mengilustrasikan alur kerja bisnis yang tidak bergantung pada kelas, aliran aktivitas dalam Use-Case, atau desain detail suatu metode	Analisis, Desain
Urutan	Memodelkan perilaku objek dalam Use-Case; fokus pada pengurutan aktivitas berdasarkan waktu	Analisis, Desain
Komunikasi	Memodelkan perilaku objek dalam Use-Case; fokus pada komunikasi antara seperangkat objek yang berkolaborasi dari suatu kegiatan	Analisis, Desain
Tinjauan Interaksi	Mengilustrasikan tinjauan aliran kontrol dari suatu proses	Analisis, Desain
Pemilihan Waktu	Menggambarkan interaksi antara satu set objek dan perubahan keadaan yang dilalui sepanjang waktu	Analisis, Desain
Mesin Status Perilaku	Memeriksa perilaku dari satu kelas	Analisis, Desain

Mesin Status Protokol	Mengilustrasikan ketergantungan antarmuka dari suatu kelas yang berbeda	Analisis, Desain
Use-Case	Menentukan kebutuhan bisnis untuk sistem dan menggambarkan interaksi antara sistem dan lingkungannya	Analisis

GAMBAR 1-18 Ringkasan Diagram UML 2.5

Diagram yang berbeda memainkan peran yang lebih penting, tergantung di mana letaknya dalam proses pengembangan sistem. Dalam beberapa kasus, teknik diagram yang sama dapat digunakan selama proses pengembangan. Dalam hal ini, diagram dimulai dengan sangat konseptual dan abstrak. Saat sistem dikembangkan, diagram berkembang untuk memasukkan detail yang pada akhirnya mengarah pada pembuatan dan pengembangan kode. Dengan kata lain, diagram mendokumentasikan persyaratan lalu menata desain. Secara keseluruhan, notasi yang konsisten, integrasi antar teknik diagram, dan penerapan diagram di seluruh proses pengembangan membuat UML menjadi bahasa yang kuat dan fleksibel untuk analis dan developer. Bab selanjutnya akan menunjukkan detail lebih lanjut mengenai penggunaan subset UML dalam analisis dan desain sistem berorientasi objek. Secara khusus, bab-bab ini menjelaskan mengenai kegiatan, Use-Case, kelas, objek, urutan, komunikasi, paket, dan diagram penyebaran dan mesin status perilaku. Kami juga memperkenalkan diagram UML opsional, yaitu diagram navigasi windows yang merupakan perpanjangan dari mesin status perilaku yang digunakan untuk merancang navigasi pengguna melalui antarmuka pengguna sistem informasi.

1.10 MENERAPKAN KONSEP PADA PATTERSON SUPERSTORE

Pelajaran ini akan memperkenalkan mengenai beberapa konsep baru tentang analisis dan desain berorientasi objek. Agar konsep-konsep ini lebih relevan dan dapat dipahami, kami akan menerapkan konsep-konsep tersebut, yang diperkenalkan di setiap bab, ke suatu perusahaan fiktif bernama Patterson Superstore.

Patterson adalah jaringan ritel yang didirikan di Pittsburgh, PA, pada tahun 1985. Saat ini, Patterson menggunakan aplikasi seluler untuk memfasilitasi pemesanan resep obat, pemberitahuan, dan layanan pengisian ulang otomatis. Layanan ini banyak digunakan oleh basis klien Patterson, dan Patterson telah memanfaatkan aplikasi seluler untuk mengungguli pesaingnya yang kurang maju secara teknis.

Sekarang kliennya ingin menggunakan teknologi ini untuk mengakses layanan klinik kesehatan. Wakil Presiden Pharmacy Services, Max Ross, menggunakan kesempatan ini untuk memposisikan Patterson sebagai pemimpin dalam hal penggunaan teknologi untuk akses klinik. Sistem yang ia harapkan dapat memungkinkan komunikasi real-time dengan tenaga medis (audio, video, dan teks), penjadwalan janji temu secara seluler, penilaian telehealth, dan diagnosis masalah kecil melalui panggilan video. Sepanjang buku ini, kita akan membahas Patterson Superstore di setiab bab untuk melihat bagaimana konsep yang diperkenalkan di setiap bab dapat mempengaruhi proyek ini.

Pertanyaan

1. Bandingkan antara fase, langkah, teknik, dan hasil.

2. Jelaskan fase-fase utama dalam SDLC.
3. Jelaskan langkah-langkah utama dalam fase perencanaan. Apa kiriman utamanya?
4. Jelaskan langkah-langkah utama dalam fase analisis. Apa kiriman utamanya?
5. Jelaskan langkah-langkah utama dalam fase desain. Apa kiriman utamanya?
6. Jelaskan langkah-langkah utama dalam fase implementasi. Apa kiriman utamanya?
7. Apa peran sponsor proyek dan komite persetujuan?
8. Apa arti penyempurnaan bertahap dalam konteks SDLC?
9. Bandingkan dan kontraskan antara metodologi yang berpusat pada proses dengan metodologi yang berpusat pada data.
10. Bandingkan dan kontraskan antara metodologi berbasis desain terstruktur secara umum dengan metodologi berbasis RAD secara umum.
11. Bandingkan dan kontraskan antara pemrograman ekstrem dan prototipe sekali pakai.
12. Jelaskan elemen utama dan masalah dengan pengembangan water-fall.
13. Jelaskan elemen utama dan isu-isu dengan pengembangan paralel.
14. Jelaskan elemen utama dan masalah dengan pengembangan bertahap.
15. Jelaskan elemen utama dan masalah dengan prototyping.
16. Jelaskan elemen utama dan masalah dengan prototipe sekali pakai.
17. Jelaskan elemen utama dan masalah dengan XP.
18. Jelaskan elemen utama dan masalah dengan Scrum.
19. Apa faktor utama dalam memilih metodologi?
20. Apa peran utama yang dimainkan oleh seorang analis sistem pada tim proyek?
21. Bandingkan dan kontraskan antara peran analis sistem, analis bisnis, dan analis infrastruktur.
22. Apa perbedaan antara kelas dan objek?
23. Apa itu metode dan pesan?
24. Mengapa enkapsulasi dan informasi menyembunyikan karakteristik penting dari sistem berorientasi objek?
25. Apa yang dimaksud dengan polimorfisme yang diterapkan pada sistem berorientasi objek?
26. Bandingkan dan kontraskan antara pengikatan dinamis dan statis.
27. Apa itu Use-Case?
28. Apa yang dimaksud dengan use-case driven?
29. Apa itu Bahasa Pemodelan Terpadu?
30. Siapa Grup Manajemen Objek?
31. Apa tujuan utama dari diagram struktur? Berikan beberapa contoh diagram struktur.
32. Untuk apa diagram perilaku digunakan? Berikan beberapa contoh diagram perilaku.
33. Mengapa pendekatan OOSAD penting untuk menjadi arsitektur-sentris?
34. Apa artinya pendekatan OOSAD menjadi inkremental dan berulang?
35. Apa saja fase dan alur kerja Proses Terpadu?
36. Bandingkan fase-fase Proses Terpadu dengan fase-fase model water-fall.
37. Fase mana dalam SDLC yang paling penting? Mengapa?
38. Jelaskan elemen utama dan isu-isu dengan pendekatan berorientasi objek untuk mengembangkan sistem informasi.

Latihan

A. Misalkan saja, Anda adalah seorang manajer proyek yang menggunakan metodologi berbasis pengembangan water-fall pada proyek besar dan kompleks. Manajer Anda baru saja membaca artikel terbaru di Computerworld yang menganjurkan penggantian metodologi ini dengan pembuatan prototipe dan mendatangi Anda untuk meminta Anda beralih. Apa yang akan Anda sampaikan?
B. Jenis dasar metodologi yang dibahas dalam bab ini dapat digabungkan dan diintegrasikan untuk membentuk metodologi hibrida baru. Misalkan Anda

menggabungkan prototipe sekali pakai dengan penggunaan pengembangan water-fall. Seperti apa metodologinya? Buatlah sebuah gambar (mirip dengan yang ada pada Gambar 1–2 hingga 1–7). Bagaimana metodologi baru ini dibandingkan dengan yang lain?

C. Cari di Web untuk berbagai jenis peluang kerja yang tersedia untuk orang-orang yang menginginkan posisi analis? Bandingkan dan kontraskan keterampilan yang diminta iklan dengan keterampilan yang kami sajikan dalam bab ini.
D. Pikirkan tentang posisi analis ideal Anda. Tulis iklan untuk mempekerjakan seseorang untuk posisi itu. Persyaratan apa yang akan dimiliki pekerjaan itu? Keterampilan dan pengalaman apa yang dibutuhkan? Bagaimana pelamar dapat menunjukkan memiliki keterampilan dan pengalaman yang sesuai?
E. Gunakan mesin pencari Web favorit Anda, temukan deskripsi alternatif dari karakteristik dasar sistem berorientasi objek.
F. Cari pemrograman berorientasi objek di Wikipedia. Tulislah laporan singkat berdasarkan entrinya.
G. Pilih bahasa pemrograman berorientasi objek, seperti C++, Java, Objective-C, Smalltalk, atau VB.Net, dan gunakan Web untuk mengetahui bagaimana bahasa tersebut mendukung karakteristik dasar sistem berorientasi objek.
H. Asumsikan bahwa Anda telah diberi tugas untuk membuat sistem berorientasi objek yang dapat digunakan untuk mendukung siswa dalam menemukan apartemen yang sesuai untuk tinggal di semester berikutnya. Apa saja jenis objek yang berbeda (yaitu, kelas) yang ingin Anda sertakan dalam sistem Anda? Atribut atau metode apa yang ingin Anda sertakan dalam definisinya? Apakah mungkin untuk mengaturnya ke dalam hierarki pewarisan? Jika demikian, lakukanlah. Jika tidak, mengapa tidak?
I. Buat hierarki pewarisan yang dapat digunakan untuk mewakili kelas berikut: akuntan, pelanggan, departemen, karyawan, manajer, organisasi, dan tenaga penjualan.
J. Selidiki Proses Terpadu Rasional (RUP) IBM di Web. RUP adalah versi komersial yang memperluas aspek Proses Terpadu. Tulis memo singkat yang menjelaskan bagaimana hal itu terkait dengan Proses Terpadu seperti yang dijelaskan dalam bab ini. (Petunjuk: Situs web yang bagus untuk memulai adalah www-01.ibm.com/software/rational/rup/.)
K. Misalkan Anda adalah seorang manajer proyek yang biasanya telah menggunakan metodologi berbasis pengembangan water-fall pada proyek besar dan kompleks. Manajer Anda baru saja membaca artikel terbaru di Computerworld yang menganjurkan penggantian metodologi ini dengan Proses Terpadu dan datang kepada Anda meminta Anda untuk beralih. Apa yang kamu katakan?
L. Misalkan Anda adalah seorang analis yang bekerja untuk sebuah perusahaan kecil untuk mengembangkan sistem akuntansi. Apakah Anda akan menggunakan Proses Terpadu untuk mengembangkan sistem, atau Anda lebih suka salah satu pendekatan lain? Mengapa?
M. Misalkan Anda adalah seorang analis yang mengembangkan sistem informasi baru untuk mengotomatisasi transaksi penjualan dan mengelola inventaris untuk setiap toko ritel dalam rantai besar. Sistem akan dipasang di setiap toko dan bertukar data dengan komputer mainframe di kantor pusat perusahaan. Apakah Anda akan menggunakan Proses Terpadu untuk mengembangkan sistem, atau Anda lebih suka salah satu pendekatan lain? Mengapa?
N. Misalkan Anda adalah seorang analis yang bekerja untuk sebuah perusahaan kecil untuk mengembangkan sistem akuntansi. Apa jenis metodologi yang akan Anda gunakan? Mengapa?
O. Misalkan Anda adalah seorang analis yang mengembangkan sistem informasi eksekutif baru yang dimaksudkan untuk memberikan informasi strategis utama dari

database perusahaan yang ada kepada eksekutif senior untuk membantu pengambilan keputusan mereka. Apa jenis metodologi yang akan Anda gunakan? Mengapa?

P. Selidiki Bahasa Pemodelan Terpadu di Web. Tulis sebuah paragraf berita singkat yang menjelaskan keadaan UML saat ini. (Petunjuk: Situs web yang bagus untuk memulai adalah www.uml.org.)

Q. Selidiki Grup Manajemen Objek (OMG) di Web. Tulis laporan yang menjelaskan tujuan OMG dan apa yang terlibat selain UML. (Petunjuk: Situs web yang bagus untuk memulai adalah www.omg.org.)

R. Menggunakan Web, temukan seperangkat CASE tool yang mendukung UML. Beberapa contoh termasuk Poseidon, Rational Rose, dan Visual Paradigm. Temukan setidaknya dua lagi. Tulis laporan singkat yang menjelaskan seberapa baik mereka mendukung UML, dan buat rekomendasi mana yang menurut Anda paling baik digunakan oleh tim proyek dalam mengembangkan sistem informasi berorientasi objek menggunakan UML.

Minicase

1. Arifwandi, Direktur Human Capital Management (HCM) di Perusahaan TELKOM Indonesia, meminta departemen SI mengembangkan manajemen tenaga penjualan dan sistem pelacakan yang memungkinkannya memantau kinerja staf penjualannya dengan lebih baik. Sayangnya, karena tumpukan besar pekerjaan yang dihadapi departemen IS, permintaannya mendapat prioritas rendah. Setelah enam bulan tidak bertindak oleh departemen IS, Arifwandi memutuskan untuk mengambil tindakan sendiri. Berdasarkan saran dari teman-temannya, Arifwandi membeli perangkat lunak database sederhana dan membangun sistem manajemen dan pelacakan tenaga penjualan sendiri.

   Meskipun sistem Arifwandi telah "selesai" selama sekitar enam minggu, masih banyak fitur yang tidak berfungsi dengan benar, dan beberapa fungsi penuh dengan kesalahan. Asisten Arifwandi sangat tidak percaya pada sistem sehingga dia diam-diam kembali menggunakan sistem berbasis kertas lamanya, karena jauh lebih dapat diandalkan.

   Saat makan malam pada suatu malam, Arifwandi mengeluh kepada seorang teman analis sistem, “Saya tidak tahu apa yang salah dengan proyek ini. Tampaknya cukup sederhana bagi saya. Orang-orang IS itu ingin saya mengikuti serangkaian langkah dan tugas yang rumit ini, tetapi saya tidak berpikir semua itu benar-benar diterapkan pada sistem berbasis PC. Saya hanya berpikir saya bisa membangun sistem ini dan mengubahnya sampai saya mendapatkan apa yang saya inginkan tanpa semua keributan dan kerumitan metodologi yang didorong oleh orang-orang IS. Maksud saya, bukankah itu hanya berlaku untuk sistem mereka yang besar dan mahal?”

   Dengan asumsi Anda adalah teman analis sistem Arifwandi, bagaimana Anda menanggapi keluhannya?

2. David, manajer proyek IS di PT. Prudential Life Assurance, sedang meninjau pengaturan kepegawaian untuk proyek besar berikutnya, pengembangan asisten penjamin emisi berbasis sistem pakar. Sistem baru ini akan melibatkan cara baru bagi penjamin emisi untuk melakukan tugas mereka. Sistem asisten penjamin emisi akan berfungsi sebagai semacam pengawas penjaminan emisi, meninjau elemen utama dari setiap aplikasi, memeriksa konsistensi dalam keputusan penjamin emisi, dan memastikan bahwa tidak ada faktor penting yang terlewatkan. Tujuan dari sistem baru ini adalah untuk meningkatkan kualitas keputusan penjamin emisi dan meningkatkan produktivitas penjamin emisi. Diharapkan bahwa sistem baru akan secara substansial mengubah cara staf penjamin emisi melakukan pekerjaan mereka.

David kecewa mengetahui bahwa karena keterbatasan anggaran, ia harus memilih antara satu dari dua anggota staf yang tersedia. Andik telah memiliki banyak pengalaman dan pelatihan dalam perilaku individu dan organisasi. Andik telah mengerjakan beberapa proyek lain di mana pengguna akhir harus membuat penyesuaian signifikan pada sistem baru, dan Andik tampaknya memiliki kemampuan untuk mengantisipasi masalah dan memperlancar transisi ke lingkungan kerja baru. David berharap ada keterlibatan Andik dalam proyek ini.

Anggota staf potensial David lainnya adalah Kenny. Sebelum bergabung dengan Prudential, Kenny memiliki pengalaman kerja yang cukup besar dengan teknologi sistem pakar yang telah dipilih Prudential untuk proyek sistem pakar ini. David mengandalkan Kenny untuk membantu mengintegrasikan teknologi sistem pakar baru ke dalam lingkungan sistem Prudential, dan juga untuk memberikan pelatihan dan wawasan di tempat kerja kepada pengembang lain di tim ini.

Mengingat anggaran David hanya akan memungkinkan dia untuk menambahkan Andik atau Kenny ke tim proyek ini, tetapi tidak keduanya, pilihan apa yang Anda rekomendasikan untuknya? Justifikasi jawaban Anda.

3. Joe Brown, presiden Roanoke Manufacturing, meminta Jack Jones, manajer departemen MIS, menyelidiki kelayakan penjualan produk mereka melalui Web. Saat ini, departemen MIS masih menggunakan mainframe IBM sebagai lingkungan penyebaran utama mereka. Sebagai langkah pertama, Jack menghubungi teman-temannya di IBM untuk mengetahui apakah mereka memiliki saran tentang bagaimana Roanoke Manufacturing dapat bergerak ke arah mendukung penjualan dalam lingkungan perdagangan elektronik sambil mempertahankan mainframe mereka sebagai sistem utama mereka. Teman-temannya menjelaskan bahwa IBM (www.ibm.com) sekarang mendukung Java dan Linux pada mainframe mereka. Jack juga mengetahui bahwa IBM memiliki Rational (www-01.ibm.com/software/rational/), pencipta UML dan Proses Terpadu. Teman-teman Jack menyarankan agar Jack menyelidiki menggunakan sistem berorientasi objek sebagai dasar untuk mengembangkan sistem baru. Mereka juga menyarankan bahwa menggunakan Rational Unified Process (RUP), Java, dan mesin Linux virtual pada mainframenya saat ini sebagai cara untuk mendukung langkah menuju sistem perdagangan elektronik terdistribusi akan melindungi investasinya saat ini dalam sistem warisannya sambil mengizinkan sistem baru. dikembangkan ke arah yang lebih modern. Meskipun teman-teman Jack di IBM sangat persuasif, Jack masih sedikit waspada untuk memindahkan operasinya dari pendekatan sistem terstruktur ke pendekatan berorientasi objek baru ini. Dengan asumsi bahwa Anda adalah salah satu teman Jack di IBM, bagaimana Anda meyakinkan dia untuk beralih menggunakan metode pengembangan sistem berorientasi objek, seperti RUP, dan menggunakan Java dan Linux sebagai dasar untuk mengembangkan dan menerapkan sistem baru di Roanoke Manufacturing saat ini. kerangka utama?

BAB 2

MANAJEMEN PROYEK

Bab ini utamanya menjelaskan mengenai alur kerja manajemen proyek dari Proses Terpadu. Langkah pertama dalam proses ini adalah mengidentifikasi proyek yang dapat memberikan nilai bagi bisnis dan membuat permintaan terhadap sistem yang menyediakan informasi dasar tentang sistem yang diusulkan. Kedua, analis melakukan analisis kelayakan untuk menentukan kelayakan teknis, ekonomi, dan organisasi dari sistem; jika sesuai, sistem dipilih dan proyek pengembangan dimulai. Ketiga, manajer proyek memperkirakan fungsionalitas proyek dan mengidentifikasi tugas-tugas yang perlu dilakukan. Keempat, manajer menyusun staf proyek. Terakhir, manajer mengidentifikasi alat, standar, dan proses yang akan digunakan; mengidentifikasi peluang penggunaan kembali; menentukan apakah proyek yang sudah ada sesuai dengan portofolio proyek yang sedang dikembangkan; dan mengidentifikasi peluang untuk memperbarui keseluruhan struktur portofolio sistem perusahaan yang saat ini digunakan.

2.1 TUJUAN

- Memahami pentingnya menghubungkan sistem informasi dengan kebutuhan bisnis.
- Mampu membuat permintaan sistem.
- Memahami cara menilai kelayakan teknis, ekonomi, dan organisasi.
- Mampu melakukan analisis kelayakan.
- Memahami bagaimana cara memilih proyek di beberapa organisasi.
- Akrab dengan struktur rincian kerja, bagan Gantt, dan diagram jaringan.
- Terbiasa dengan estimasi upaya terhadap Use-Case.
- Mampu membuat rencana kerja proyek berulang.
- Memahami cara mengelola ruang lingkup, menyempurnakan perkiraan, dan mengelola risiko proyek.
- Akrab dengan bagaimana menyusun staf proyek.
- Memahami bagaimana interaksi antara lingkungan dan alur kerja infrastruktur dengan alur kerja manajemen proyek.

2.2 PENDAHULUAN

Sebagian besar proyek yang ada di kehidupan manusia, seperti pernikahan atau perayaan kelulusan, memerlukan perencanaan dan pengelolaan. Berbulan-bulan sebelumnya dihabiskan untuk mengidentifikasi dan melakukan semua tugas yang perlu diselesaikan, seperti mengirim undangan dan memilih menu, dan waktu serta uang dialokasikan secara hati-hati dalam prosesnya. Sepanjang perjalanan, keputusan dicatat, masalah ditangani, dan perubahan dibuat. Meningkatnya popularitas jasa penyelenggara pesta, seseorang yang tugasnya mengoordinasikan sebuah pesta, menunjukkan betapa sulitnya pekerjaan ini. Pada akhirnya, keberhasilan setiap pihak sangat berkaitan dengan upaya yang dilakukan dalam perencanaan. Proyek pengembangan sistem bisa jauh lebih rumit daripada proyek yang kita temui dalam kehidupan sehari-hari kita—biasanya, lebih banyak orang yang terlibat (misalnya, organisasi), biayanya juga lebih tinggi, dan lebih banyak tugas yang harus diselesaikan. Karena kompleksitas perangkat lunak dan pengembangannya, hampir tidak mungkin untuk "mengetahui" semua hal yang mungkin terjadi selama proyek pengembangan sistem. Oleh karena itu, tidak heran bahwa "jasa penyelenggara pesta" ada dalam suatu proyek sistem informasi: Mereka disebut dengan manajer proyek.

Manajemen proyek adalah proses perencanaan dan pengendalian pengembangan suatu sistem pada waktu tertentu dengan biaya minimum dan fungsionalitas yang tepat.[25] Secara umum, proyek adalah serangkaian kegiatan dengan titik awal dan titik akhir yang dimaksudkan untuk menciptakan sistem yang dapat memberi nilai bagi bisnis. Seorang manajer proyek memiliki tanggung jawab utama untuk mengelola ratusan tugas dan peran yang perlu dikoordinasikan dengan hati-hati. Saat ini, manajemen proyek adalah profesi yang sebenarnya, dan analis menghabiskan waktu bertahun-tahun untuk mengerjakan proyek sebelum akhirnya manajemen mengelolanya. Namun, dalam banyak kasus, tuntutan yang tidak masuk akal yang ditetapkan oleh sponsor proyek dan manajer bisnis mengakibatkan manajemen proyek menjadi sangat sulit. Seringnya, mendekati musim liburan, kesempatan untuk memenangkan proposal dengan tawaran rendah, atau peluang pendanaan dapat menekan manajer proyek agar mampu memberi penawaran sistem jauh sebelum mereka dapat mengirimkannya. Jadwal yang terlalu optimis ini dianggap sebagai salah satu masalah terbesar yang dihadapi proyek; bukannya mendorong proyek ke depan lebih cepat, hal ini justru menyebabkan penundaan. Sumber lain adalah perubahan sifat teknologi informasi. Suatu inovasi dalam teknologi informasi mungkin terlihat sangat menarik sehingga organisasi menerima proyek yang menggunakan teknologi ini tanpa menilai apakah teknologi tersebut meningkatkan nilai organisasi; alih-alih, teknologi itu sendirilah yang kelihatannya penting. Permasalahan biasanya dapat ditelusuri kembali ke awal pengembangan sistem, di mana perhatian yang diberikan untuk mengidentifikasi nilai bisnis terlalu sedikit dan pemahaman risiko yang terkait dengan proyek juga kurang.

Selama fase awal Proses Terpadu dari suatu proyek pengembangan sistem baru, seorang—manajer, anggota staf, perwakilan penjualan, atau analis sistem—biasanya mengidentifikasi beberapa nilai bisnis yang dapat diperoleh dari penggunaan teknologi informasi. Proyek pengembangan sistem baru harus dimulai berdasarkan kebutuhan atau peluang bisnis. Banyak ide untuk sistem baru atau perbaikan sistem berawal dari penerapan teknologi baru, tetapi pemahaman tentang teknologi biasanya dari pemahaman yang kuat mengenai bisnis dan tujuannya. Hal ini bukan berarti bahwa orang teknis tidak boleh merekomendasikan proyek sistem baru. Faktanya, situasi yang ideal adalah orang-orang TI (yaitu, para ahli dalam sistem) dan orang-orang bisnis (yaitu, para ahli dalam bisnis) bekerja sama dalam mengembangkan teknologi untuk mendukung kebutuhan bisnis. Dengan cara ini, organisasi dapat memanfaatkan teknologi inovatif yang tersedia sambil memastikan bahwa proyek benar-benar didasarkan pada tujuan bisnis nyata, seperti meningkatkan penjualan, meningkatkan layanan pelanggan, dan mengurangi biaya operasional. Pada akhirnya, sistem informasi harus mempengaruhi organisasi (dengan cara yang positif!). Untuk memastikan bahwa kebutuhan bisnis nyata benar-benar sedang ditangani, organisasi bisnis yang terpengaruh (disebut sponsor proyek), mengusulkan proyek pengembangan sistem baru menggunakan permintaan sistem. Permintaan sistem secara efektif memulai fase awal untuk proyek pengembangan sistem baru. Permintaan diteruskan ke komite persetujuan untuk dipertimbangkan. Komite persetujuan meninjau permintaan tersebut dan membuat keputusan awal apakah akan menyelidiki proposal tersebut atau tidak. Jika komite awalnya menyetujui

[25] Untuk mengetahui secara komprehensif mengenai manajemen proyek untuk sistem informasi, baca R.K. Wysocki, Effective Project Management: Traditional, Agile, Extreme, 5th Ed. (Indianapolis, IN: Wiley Publishing, 2009). Selain itu, Project Management Institute (www.pmi.org) dan Information Systems Community of Practice of the Project Management Institute (is.vc.pmi.org) memiliki sumber yang bagus mengenai manajemen proyek sistem informasi. Terakhir, berikut ini adalah buku yang bagus mengenai manajemen proyek untuk proyek berorientasi objek: G. Booch, Object Solutions: Managing the Object-Oriented Project (Menlo Park, CA: Addison-Wesley, 1996); M. R. Cantor, Object-Oriented Project Management with UML (New York: Wiley, 1998); A. Cockburn, Surviving Object-Oriented Projects: A Manager's Guide (Reading, MA: Addison-Wesley, 1998); I. Jacobson, G. Booch, dan J. Rumbaugh, The Unified Software Development Process (Reading, MA: Addison-Wesley, 1999); W. Royce, Software Project Management: A Unified Framework (Reading, MA: Addison-Wesley, 1998).

permintaan tersebut, tim pengembangan sistem mengumpulkan lebih banyak informasi untuk menentukan kelayakan proyek.

Analisis kelayakan memainkan peran penting dalam memutuskan apakah akan melanjutkan proyek pengembangan sistem informasi. Analisis ini mengkaji pro dan kontra teknis, ekonomi, dan organisasi dalam mengembangkan sistem, dan memberikan gambaran yang sedikit lebih rinci kepada organisasi mengenai keuntungan berinvestasi pada sistem serta hambatan apa yang dapat muncul. Dalam kebanyakan kasus, sponsor proyek bekerja sama dengan tim pengembangan untuk mengembangkan analisis kelayakan. Setelah analisis kelayakan selesai, hasilnya diserahkan ke komite persetujuan, bersamaan dengan permintaan sistem yang direvisi. Komite kemudian memutuskan apakah akan menyetujui proyek atau menolaknya, atau tetap mengajukannya hingga informasi tambahan tersedia. Proyek dipilih dengan mempertimbangkan risiko dan return dan dengan membuat trade-off di tingkat organisasi.

Setelah komite menyetujui sebuah proyek, tim pengembangan harus merencanakan pengembangan sistem yang sebenarnya dengan hati-hati. Karena kita mengikuti pendekatan berbasis Proses Terpadu, rencana kerja pengembangan sistem akan berkembang sepanjang proses pengembangan. Dengan pendekatan evolusioner ini, salah satu faktor keberhasilan yang penting untuk manajemen proyek adalah memulai dengan penilaian realistis dari pekerjaan yang perlu diselesaikan dan kemudian mengelola proyek sesuai dengan penilaian itu. Hal ini dapat dicapai dengan membuat dan mengelola rencana kerja secara hati-hati, memperkirakan upaya untuk mengembangkan sistem, mengatur staf proyek, dan mengoordinasikan kegiatan proyek.

Selain membahas materi di atas, bab ini juga mencakup tiga alat manajemen proyek tradisional yang sangat berguna untuk mengelola proyek pengembangan sistem berorientasi objek, yaitu struktur rincian kerja, bagan Gantt, dan diagram jaringan.

2.3 IDENTIFIKASI PROYEK

Sebuah proyek diidentifikasi ketika seseorang dalam organisasi mengidentifikasi kebutuhan bisnis untuk membangun sebuah sistem. Hal ini dapat terjadi di dalam unit bisnis atau TI, yang berasal dari komite pengarah yang bertugas mengidentifikasi peluang bisnis, atau yang merupakan pengembangan dari rekomendasi yang dibuat oleh konsultan eksternal. Contoh kebutuhan bisnis yaitu sesuatu yang mendukung kampanye pemasaran baru, menjangkau jenis pelanggan baru, atau yang meningkatkan interaksi dengan pemasok. Kadang-kadang, kebutuhan muncul dari semacam "rasa sakit" dalam organisasi, seperti penurunan pangsa pasar, tingkat layanan pelanggan yang buruk, atau persaingan yang meningkat. Di lain waktu, inisiatif dan strategi bisnis baru dibuat, dan sistem diperlukan untuk mewujudkannya.

Kebutuhan bisnis juga dapat muncul ketika organisasi mengidentifikasi cara unik dan kompetitif dalam menggunakan TI. Banyak organisasi mengawasi teknologi yang muncul, yaitu teknologi yang masih dikembangkan dan belum layak untuk penggunaan bisnis secara luas. Misalnya, jika perusahaan tetap mengikuti perkembangan teknologi seperti augmented reality, game, kartu pintar, dan perangkat seluler, mereka dapat mengembangkan strategi bisnis yang memanfaatkan kemampuan teknologi ini dan memperkenalkannya ke pasar sebagai penggerak pertama. Idealnya, mereka dapat memanfaatkan keuntungan penggerak pertama ini dengan menghasilkan uang dan terus berinovasi sementara pesaing tertinggal di belakang.

Sponsor proyek adalah seseorang yang menyadari kebutuhan bisnis yang kuat untuk sebuah sistem dan memiliki minat untuk melihat keberhasilan sistem. Dia akan bekerja

sepanjang proses pengembangan untuk memastikan bahwa proyek bergerak ke arah yang benar dari perspektif bisnis. Sponsor proyek berfungsi sebagai kontak utama sistem. Biasanya, sponsor proyek berasal dari fungsi bisnis, seperti pemasaran, akuntansi, atau keuangan; namun, anggota TI juga dapat menjadi sponsor atau co-sponsor proyek.

Ukuran atau ruang lingkup proyek menentukan jenis sponsor yang dibutuhkan. Sebuah sistem departemen kecil mungkin memerlukan sponsor dari hanya satu manajer, sedangkan organisasi yang besar mungkin memerlukan dukungan dari seluruh tim manajemen senior dan bahkan CEO. Jika sebuah proyek murni bersifat teknis (misalnya, perbaikan infrastruktur TI atau riset mengenai kelayakan teknologi yang muncul), maka sponsor dari TI adalah yang paling tepat. Ketika proyek sangat penting bagi bisnis namun secara teknis kompleks, sponsor gabungan dari bisnis dan TI mungkin diperlukan.

Kebutuhan bisnis harus mendorong kebutuhan bisnis tingkat tinggi bagi sistem. Persyaratan adalah apa yang akan dilakukan sistem informasi, atau fungsionalitas yang akan dimilikinya. Persyaratan ini perlu dijelaskan pada tingkat tinggi sehingga komite persetujuan dan tim proyek memahami apa yang diharapkan oleh bisnis dari suatu produk akhir. Persyaratan bisnis adalah fitur dan kemampuan yang harus disertakan oleh sistem informasi, seperti kemampuan untuk mengumpulkan pesanan pelanggan secara online atau kemampuan pemasok untuk menerima informasi persediaan saat pesanan diterima dan penjualan dilakukan.

Sponsor proyek juga harus memiliki gagasan tentang nilai bisnis yang akan diperoleh dari sistem, baik secara nyata maupun tidak nyata. Nilai nyata dapat diukur dan diukur dengan mudah (misalnya, pengurangan 2 persen dalam biaya operasi). Nilai tidak nyata dihasilkan dari keyakinan intuitif bahwa sistem memberikan manfaat penting bagi organisasi tetapi sulit diukur (misalnya, peningkatan layanan pelanggan atau posisi kompetitif yang lebih baik).

Setelah sponsor proyek mengidentifikasi proyek yang memenuhi kebutuhan penting suatu bisnis dan dia dapat mengidentifikasi persyaratan dan nilai bisnis sistem, maka saatnya untuk memulai proyek secara resmi. Di sebagian besar organisasi, inisiasi proyek dimulai dengan dokumen yang disebut permintaan sistem.

Permintaan Sistem

Permintaan sistem adalah dokumen yang menjelaskan mengenai alasan bisnis untuk membangun sistem dan nilai yang akan diberikan oleh sistem. Sponsor proyek biasanya melengkapi formulir ini sebagai bagian dari proses pemilihan proyek sistem formal dalam organisasi. Umumnya, permintaan sistem mencakup lima elemen: sponsor proyek, kebutuhan bisnis, persyaratan bisnis, nilai bisnis, dan masalah khusus. Sponsor menjelaskan tentang siapa yang akan menjadi kontak utama untuk proyek tersebut, dan kebutuhan bisnis menyajikan alasan yang mendorong proyek tersebut. Persyaratan bisnis suatu proyek mengacu pada kemampuan bisnis yang harus dimiliki sistem, dan nilai bisnis menggambarkan manfaat yang diharapkan organisasi dari sistem. Isu-isu khusus dimasukkan dalam dokumen sebagai informasi lain yang harus dipertimbangkan dalam menilai proyek. Misalnya, proyek mungkin perlu diselesaikan dengan tenggat waktu tertentu. Tim proyek harus mempertimbangkan keadaan khusus yang dapat memengaruhi hasil sistem. Gambar 2-1 menunjukkan template untuk permintaan sistem.

Permintaan sistem yang telah selesai diajukan ke komite persetujuan untuk dipertimbangkan. Komite persetujuan bisa jadi merupakan komite pengarah perusahaan yang mengadakan pertemuan secara teratur untuk membuat keputusan sistem informasi, atau seorang eksekutif senior yang memiliki kendali atas sumber daya organisasi, atau badan pembuat keputusan lainnya yang mengatur penggunaan investasi bisnis. Komite meninjau permintaan sistem dan membuat keputusan awal berdasarkan informasi yang diberikan,

apakah akan menyelidiki proposal atau tidak. Setelah itu, langkah selanjutnya adalah melakukan analisis kelayakan.

Permintaan Sistem—Nama Proyek	
Sponsor Proyek:	Nama sponsor proyek
Kebutuhan Bisnis:	Deskripsi singkat kebutuhan bisnis
Persyaratan Bisnis:	Deskripsi persyaratan bisnis
Nilai Bisnis:	Nilai yang diharapkan yang dapat diberikan oleh sistem
Masalah Khusus atau Kendala:	Setiap informasi tambahan yang mungkin relevan dengan pemegang kepentingan

GAMBAR 2-1 Template Permintaan Sistem

2.4 ANALISIS KELAYAKAN

Setelah kebutuhan sistem dan kebutuhan bisnis ditentukan, maka saatnya untuk membuat kasus bisnis yang lebih rinci untuk lebih memahami peluang dan keterbatasan yang terkait dengan proyek yang diusulkan. Analisis kelayakan memandu organisasi dalam menentukan apakah akan melanjutkan proyek atau tidak. Analisis kelayakan juga berguna untuk mengidentifikasi risiko penting yang terkait dengan proyek yang harus ditangani jika proyek disetujui. Seperti halnya permintaan sistem, setiap organisasi memiliki proses dan format sendiri untuk analisis kelayakan, tetapi sebagian besar mencakup tiga jenis: kelayakan teknis, kelayakan ekonomi, dan kelayakan organisasi. Hasil analisis ini digabungkan menjadi studi kelayakan yang diajukan kepada komite persetujuan (lihat Gambar 2-2).

Meskipun sekarang kita membahas analisis kelayakan dalam konteks untuk memulai proyek, sebagian besar tim proyek akan merevisi studi kelayakan mereka selama proses pengembangan dan meninjau kembali isinya di berbagai pos pemeriksaan sepanjang proyek. Jika suatu saat risiko dan keterbatasan proyek lebih besar daripada manfaatnya, tim proyek dapat memutuskan untuk membatalkan proyek atau melakukan perbaikan.

Kelayakan Teknis

Jenis analisis kelayakan yang pertama membahas mengenai kelayakan teknis proyek, yaitu sejauh mana sistem dapat berhasil dirancang, dikembangkan, dan dipasang oleh grup TI. nalisis kelayakan teknis pada dasarnya adalah analisis risiko teknis yang berusaha menjawab pertanyaan: “Bisakah kita membangunnya?”[26]

Kelayakan Teknis: Bisakah Kita Membangunnya?

- Familier dengan area Fungsional: Kurang familier menghasilkan lebih banyak risiko
- Familier dengan Teknologi: Kurang familier menghasilkan lebih banyak risiko
- Ukuran Proyek: Proyek besar memiliki lebih banyak risiko

[26] Kami menggunakan istilah ‘membangun’ dalam arti luas. Organisasi juga dapat memilih untuk membeli paket perangkat lunak komersial dan menginstalnya, dalam hal ini, pertanyaannya mungkin: Bisakah kita memilih paket yang tepat dan berhasil menginstalnya?

- Kompatibilitas: Semakin sulit untuk mengintegrasikan sistem dengan teknologi perusahaan yang sudah ada, semakin tinggi risikonya

Kelayakan Ekonomi: Haruskah Kita Membangunnya?

- Biaya pengembangan
- Biaya operasional tahunan
- Manfaat tahunan (penghematan biaya dan pendapatan)
- Biaya dan manfaat tak nyata

Kelayakan Organisasi: Jika Kita Membangunnya, Apakah Mereka Akan Datang?

- Apakah proyek secara strategis selaras dengan bisnis?
- Pemenang proyek
- Manajemen senior
- Pengguna
- Pemegang kepentingan lainnya

GAMBAR 2-2 Faktor Penilaian Analisis Kelayakan

Banyak risiko yang dapat membahayakan keberhasilan penyelesaian suatu proyek. Pertama adalah kurang terbiasanya pengguna dan analis terhadap area fungsional. Ketika analis tidak terbiasa dengan area fungsional bisnis, mereka berpeluang besar untuk salah paham dengan pengguna atau kehilangan peluang untuk perbaikan. Risiko meningkat secara drastis ketika pengguna sendiri kurang akrab dengan aplikasi, seperti misalnya dengan pengembangan sistem untuk mendukung inovasi bisnis. Secara umum, mengembangkan sistem baru lebih berisiko daripada mengembangkan sistem yang sudah ada karena sistem yang ada cenderung lebih mudah dipahami.

Familiaritas dengan teknologi adalah sumber penting lain dari risiko teknis. Ketika sebuah sistem menggunakan teknologi yang belum pernah digunakan sebelumnya di dalam organisasi, kemungkinan besar akan terjadi masalah dan penundaan akan terjadi karena adanya kebutuhan untuk mempelajari cara menggunakan teknologi tersebut. Risiko meningkat secara drastis ketika teknologi itu sendiri masih baru.

Ukuran proyek merupakan pertimbangan penting, apakah diukur sebagai jumlah orang dalam tim pengembangan, lamanya waktu yang dibutuhkan untuk menyelesaikan proyek, atau jumlah fitur yang berbeda dalam suatu sistem. Proyek yang lebih besar memiliki lebih banyak risiko, baik karena lebih rumit untuk dikelola atau karena adanya kemungkinan bahwa persyaratan sistem yang penting dapat terabaikan atau disalahpahami. Selanjutnya, tingkat sejauh mana proyek terintegrasi dengan sistem lain dapat menyebabkan masalah karena kompleksitas meningkat ketika sistem harus bekerja sama dengan beberapa sistem lain.

Pada akhirnya, tim proyek perlu mempertimbangkan kompatibilitas sistem baru dengan teknologi yang sudah ada di organisasi. Sistem jarang dibangun dalam ruang hampa—sistem dibangun dalam organisasi yang sudah memiliki banyak sistem. Teknologi dan aplikasi baru perlu diintegrasikan dengan lingkungan yang sudah ada karena berbagai alasan. Sistem itu mungkin mengandalkan data dari sistem yang sudah ada, karena sistem itu mungkin menyediakan data yang diperlukan aplikasi lain, dan sistem itu mungkin harus menggunakan infrastruktur komunikasi perusahaan yang sudah ada.

Penilaian kelayakan teknis proyek tidak dipotong dan dibiarkan karena dalam banyak kasus, beberapa interpretasi dari kondisi yang mendasarinya diperlukan. Salah satu pendekatannya adalah membandingkan proyek yang sedang dipertimbangkan dengan proyek sebelumnya di organisasi tsb. Pilihan lainnya adalah berkonsultasi dengan TI profesional yang berpengalaman di organisasi atau konsultan TI eksternal; sering kali mereka dapat menilai apakah suatu proyek layak dari perspektif teknis.

Kelayakan Ekonomi

Elemen kedua dari analisis kelayakan adalah melakukan analisis kelayakan ekonomi (disebut juga sebagai analisis biaya-manfaat), yang mengidentifikasi risiko keuangan yang terkait dengan proyek. Analisis ini mencoba menjawab pertanyaan, Haruskah kita membangun sistem? Kelayakan ekonomi ditentukan dengan mengidentifikasi biaya dan manfaat yang terkait dengan sistem, menilainya, dan kemudian menghitung arus kas dan laba atas investasi untuk proyek tersebut. Semakin mahal proyek, semakin teliti dan rinci analisisnya. Gambar 2-3 mencantumkan langkah-langkah dalam melakukan analisis biaya-manfaat; setiap langkah dijelaskan di bagian setelah ini.

Mengidentifikasi Biaya dan Manfaat. Tugas pertama ketika mengembangkan analisis kelayakan ekonomi adalah mengidentifikasi jenis biaya dan manfaat yang akan dimiliki sistem dan mencantumkannya di sepanjang kolom kiri spreadsheet. Gambar 2-4 mencantumkan contoh biaya dan manfaat yang mungkin disertakan.

1. Mengidentifikasi Biaya dan Manfaat	Buatlah daftar biaya dan manfaat nyata untuk proyek tersebut. Sertakan biaya sekali pakai dan biaya berulang.
2. Menetapkan Nilai untuk Biaya dan Manfaat	Bekerjalah dengan pengguna bisnis dan TI profesional untuk membuat angka untuk setiap biaya dan manfaat. Bahkan yang tidak nyata juga harus dinilai jika memungkinkan.
3. Menentukan Arus Kas	Proyeksikan biaya dan manfaatnya selama periode waktu tertentu, biasanya tiga sampai lima tahun. Terapkan tingkat pertumbuhan ke dalam angka, jika perlu.
4. Menentukan Net Present Value (NPV)	Hitung berapa nilai biaya dan manfaat di masa mendatang jika diukur dengan standar saat ini. Anda harus memilih tingkat pertumbuhan untuk menerapkan rumus NPV.
5. Menentukan Return on Investment (ROI)	Hitung berapa banyak uang yang akan diterima organisasi sebagai imbalan atas investasi menggunakan rumus ROI.
6. Menentukan Break-Even Point (Titik Impas)	Carilah tahun pertama di mana sistem memiliki manfaat lebih besar dibanding biaya. Terapkan rumus titik impas dengan menggunakan angka-angka dari tahun tersebut. Hal ini akan membantu Anda memahami berapa lama waktu yang dibutuhkan sebelum sistem memberikan nilai nyata bagi organisasi.
7. Menggambarkan Titik Impas	Plot biaya dan manfaat tahunan pada grafik dalam bentuk garis. Titik di mana garis bersilangan adalah titik impas.

GAMBAR 2-3 Langkah-Langkah Melakukan Kelayakan Ekonomi

Biaya Pengembangan	Biaya Operasional
Gaji Tim Pengembangan	Software Upgrade
Biaya Konsultan	Biaya Lisensi Software
Pelatihan Pengembangan	Biaya Perbaikan Hardware
Hardware and Software	Hardware Upgrade
Pemasangan Vendor	Gaji Tim Operasional
Ruang dan Peralatan Kantor	Biaya Komunikasi
Biaya Konversi Data	Pelatihan Pengguna
Manfaat Nyata	Manfaat Tidak Nyata
Peningkatan Gaji	Peningkatan Saham
Pengurangan Staf	Peningkatan Penerimaan Merek
Pengurangan Inventaris	Produk dengan Kualitas Tinggi
Pengurangan Biaya TI	Peningkatan Customer Service
Harga Pemasok yang Lebih Baik	Hubungan Pemasok yang Lebih Baik

GAMBAR 2-4 Contoh Biaya dan Manfaat Kelayakan Ekonomi

Biaya dan manfaat dapat dipecah menjadi empat kategori: biaya pengembangan, biaya operasional, manfaat nyata, dan tidak nyata. Biaya pengembangan adalah biaya nyata yang dikeluarkan selama pembangunan sistem, seperti gaji tim proyek, biaya perangkat keras dan perangkat lunak, biaya konsultan, pelatihan, serta ruang dan peralatan kantor. Biaya pengembangan biasanya dianggap sebagai biaya satu kali. Biaya operasional adalah biaya nyata yang diperlukan untuk mengoperasikan sistem, seperti gaji staf operasi, biaya lisensi perangkat lunak, peningkatan peralatan, dan biaya komunikasi. Biaya operasional biasanya dianggap sebagai biaya berkelanjutan.

Pendapatan dan penghematan biaya adalah manfaat nyata sistem yang mungkin diperoleh organisasi atau biaya nyata yang mungkin dapat dihindari oleh organisasi. Manfaat nyata dapat mencakup peningkatan penjualan, pengurangan staf, dan pengurangan persediaan. Tentu saja, sebuah proyek juga dapat mempengaruhi laba organisasi dengan menuai manfaat tidak nyata atau menimbulkan biaya tidak nyata. Biaya dan manfaat tidak nyata lebih sulit untuk dimasukkan ke dalam kelayakan ekonomi karena didasarkan pada intuisi dan keyakinan, bukan pada "angka yang sulit". Meskipun demikian, manfaat tsb harus terdaftar dalam spreadsheet bersama dengan barang-barang nyata lainnya.

Menetapkan Nilai untuk Biaya dan Manfaat. Setelah jenis biaya dan manfaat diidentifikasi, analis menetapkan nominal harga tertentu kepada keduanya. Hal ini mungkin tampak mustahil; bagaimana seseorang dapat menghitung biaya dan manfaat yang belum terjadi? Dan bagaimana prediksi itu bisa realistis? Meskipun tugas ini sangat sulit, analis harus melakukan yang terbaik yang mereka bisa untuk menghasilkan angka yang masuk akal untuk

semua biaya dan manfaat. Karena hanya dengan inilah komite persetujuan dapat membuat keputusan yang tentang apakah proyek akan dilanjutkan atau tidak.

Strategi terbaik untuk memperkirakan biaya dan manfaat adalah dengan mengandalkan orang-orang yang memiliki pemahaman mengenai hal itu. Misalnya, biaya dan manfaat yang terkait dengan teknologi atau proyek itu sendiri dapat disediakan oleh grup TI perusahaan atau konsultan eksternal, dan pengguna bisnis dapat mengembangkan angka yang terkait dengan bisnis (misalnya, proyeksi penjualan, tingkat pesanan). Analis juga dapat melihat pada proyek sebelumnya, laporan industri, dan informasi vendor, meskipun pendekatan ini mungkin sedikit kurang akurat. Semua perkiraan mungkin akan direvisi seiring berjalannya proyek.

Terkadang analis dapat membuat daftar manfaat tidak nyata, seperti peningkatan layanan pelanggan, tanpa menetapkan harganya, tetapi di lain waktu mereka tetap harus membuat perkiraan mengenai nilai manfaat tidak nyata itu. Jika memungkinkan, mereka harus menghitung biaya atau manfaat tidak nyata. Jika tidak, mereka tidak akan tahu apakah biaya dan manfaat tsb telah terealisasi. Pertimbangkan mengenai sistem yang seharusnya meningkatkan layanan pelanggan. Manfaat ini memang tidak nyata, tetapi asumsikan bahwa layanan pelanggan yang lebih besar dapat menurunkan jumlah keluhan pelanggan sebesar 10 persen setiap tahun selama tiga tahun dan bahwa Rp. 28.200.000.000 dihabiskan untuk biaya telepon dan operator telepon yang menangani panggilan keluhan. Tiba-tiba ada beberapa angka yang sangat nyata yang dapat digunakan untuk menetapkan tujuan dan mengukur manfaat dari yang tidak nyata.

Benefits[a]	
Increased sales	500,000
Improved customer service[b]	70,000
Reduced inventory costs	68,000
Total benefits	**638,000**
Development costs	
2 servers @ $125,000	250,000
Printer	100,000
Software licenses	34,825
Server software	10,945
Development labor	1,236,525
Total development costs	**1,632,295**
Operational costs	
Hardware	54,000
Software	20,000
Operational labor	111,788
Total operational costs	**185,788**
Total costs	**1,818,083**

Eterangan Gambar :

[a]Manfaat penting namun tidak berwujud adalah kemampuan untuk menawarkan layanan yang ditawarkan pesaing kita saat ini.

[b]Nomor layanan pelanggan didasarkan pada pengurangan biaya untuk panggilan telepon keluhan pelanggan.

GAMBAR 2-5 Menetapkan Nilai untuk Biaya dan Manfaat

Gambar 2-5 menunjukkan biaya dan manfaat dengan nilai dolar yang ditetapkan. Perhatikan bahwa manfaat tidak berwujud dari layanan pelanggan telah diukur berdasarkan sedikitnya panggilan telepon keluhan pelanggan. Manfaat tidak nyata untuk dapat menawarkan layanan yang ditawarkan pesaing saat ini tidak dihitung, tetapi terdaftar sehingga komite persetujuan akan mempertimbangkan manfaatnya ketika menilai kelayakan ekonomi sistem.

Menentukan Arus Kas. Suatu analisis biaya-manfaat formal biasanya berisi biaya dan manfaat selama beberapa tahun (biasanya tiga sampai lima tahun) untuk menunjukkan arus kas dari waktu ke waktu (lihat Gambar 2-6). Saat menggunakan metode arus kas ini, tahun dicantumkan di bagian atas spreadsheet untuk mewakili periode waktu analisis, dan nilai numerik dimasukkan ke dalam sel yang sesuai di dalam badan spreadsheet. Terkadang jumlah tetap dimasukkan ke dalam kolom. Misalnya, Gambar 2-6 mencantumkan jumlah yang sama untuk panggilan keluhan pelanggan dan biaya persediaan selama lima tahun. Biasanya jumlah ditambah dengan beberapa tingkat pertumbuhan untuk menyesuaikan dengan inflasi atau peningkatan bisnis, seperti yang ditunjukkan dengan peningkatan 6 persen yang ditambahkan ke angka penjualan dalam contoh spreadsheet. Terakhir, jumlah total ditambahkan untuk menentukan manfaat keseluruhan yang akan diperoleh; semakin tinggi total keseluruhan, semakin besar kelayakan ekonomi dari solusi.

Menentukan Net Present Value dan Return on Investment. Ada beberapa masalah dengan metode arus kas—(1) metode ini tidak mempertimbangkan nilai waktu terhadap uang (yaitu, satu dolar hari ini tidak bernilai satu dolar besok), dan (2) metode ini tidak menunjukkan keseluruhan "bang for the buck" yang diterima organisasi dari investasinya. Oleh karena itu, beberapa tim proyek menambahkan perhitungan tambahan ke spreadsheet untuk memberikan gambaran yang lebih akurat tentang nilai proyek kepada komite persetujuan.

Net present value (NPV) digunakan untuk membandingkan nilai sekarang dari arus kas masa mendatang dengan pengeluaran investasi yang diperlukan untuk melaksanakan proyek. Misalnya, jika Anda memiliki teman yang berhutang satu dolar hari ini tetapi malah memberi Anda satu dolar tiga tahun dari sekarang. Mengingat peningkatan nilai 10 persen, Anda akan menerima setara dengan 75 sen dalam persyaratan hari ini.

NPV dapat dihitung dengan berbagai cara, beberapa di antaranya sangat kompleks. Gambar 2-7 menunjukkan perhitungan dasar yang dapat digunakan dalam analisis arus kas Anda untuk mendapatkan nilai yang lebih relevan. Pada Gambar 2-6, nilai sekarang dari biaya dan manfaat dihitung terlebih dahulu (yaitu, ditampilkan dengan tarif diskon). Kemudian, NPV dihitung, dan hasilnya menunjukkan tingkat diskon dari gabungan biaya dan manfaat.

	2015	2016	2017	2018	2019	Total
Increased sales	500,000	530,000	561,800	595,508	631,238	
Reduction in customer complaint calls	70,000	70,000	70,000	70,000	70,000	
Reduced inventory costs	68,000	68,000	68,000	68,000	68,000	
TOTAL BENEFITS:	638,000	668,000	699,800	733,508	769,238	
PV OF BENEFITS:	**619,417**	**629,654**	**640,416**	**651,712**	**663,552**	**3,204,752**
PV OF ALL BENEFITS:	**619,417**	**1,249,072**	**1,889,488**	**2,541,200**	**3,204,752**	
2 Servers @ $125,000	250,000	0	0	0	0	
Printer	100,000	0	0	0	0	
Software licenses	34,825	0	0	0	0	
Server software	10,945	0	0	0	0	
Development labor	1,236,525	0	0	0	0	
TOTAL DEVELOPMENT COSTS:	1,632,295	0	0	0	0	
Hardware	54,000	81,261	81,261	81,261	81,261	
Software	20,000	20,000	20,000	20,000	20,000	
Operational labor	111,788	116,260	120,910	125,746	130,776	
TOTAL OPERATIONAL COSTS:	185,788	217,521	222,171	227,007	232,037	
TOTAL COSTS:	1,818,083	217,521	222,171	227,007	232,037	
PV OF COSTS:	**1,765,129**	**205,034**	**203,318**	**201,693**	**200,157**	**2,575,331**
PV OF ALL COSTS:	**1,765,129**	**1,970,163**	**2,173,481**	**2,375,174**	**2,575,331**	
TOTAL PROJECT BENEFITS COSTS:	**(1,180,083)**	**450,479**	**477,629**	**506,501**	**537,201**	
YEARLY NPV:	**(1,145,712)**	**424,620**	**437,098**	**450,019**	**463,395**	**629,421**
CUMULATIVE NPV:	**(1,145,712)**	**(721,091)**	**(283,993)**	**166,026**	**629,421**	
RETURN ON INVESTMENT:	**24.44%**	(629,421/2,575,331)				
BREAK-EVEN POINT:	**3.63 years**	[break-even occurs in year 4; (450,019 − 166,026)/450,019 = 0.63]				
INTANGIBLE BENEFITS:	This service is currently provided by competitors Improved customer satisfaction					

GAMBAR 2-6 Analisis Biaya-Manfaat

Return on Investment (ROI) adalah perhitungan yang tercantum di spreadsheet yang mengukur jumlah uang yang diterima organisasi sebagai imbalan atas uang yang dibelanjakannya. Hasil ROI tinggi ketika manfaat jauh lebih besar daripada biaya. ROI ditentukan dengan mencari total manfaat dikurangi biaya sistem dan membagi angka tersebut dengan total biaya sistem (lihat Gambar 2-7). ROI dapat ditentukan per tahun atau per proyek selama periode waktu tertentu. Salah satu kelemahan ROI adalah hanya mempertimbangkan titik akhir investasi, bukan arus kas yang terjadi di antaranya, sehingga tidak boleh digunakan sebagai satu-satunya indikator nilai proyek. Spreadsheet pada Gambar 2-6 menunjukkan angka ROI.

Menentukan Titik Impas. Jika tim proyek perlu melakukan analisis biaya-manfaat yang ketat, mungkin perlu memasukkan informasi tentang lamanya waktu sebelum proyek mencapai titik impas, atau kapan pengembalian akan sesuai dengan jumlah yang diinvestasikan dalam proyek. Semakin lama waktu yang dibutuhkan untuk mencapai titik impas, semakin berisiko proyek tersebut. Titik impas ditentukan dengan melihat arus kas dari waktu ke waktu dan mengidentifikasi tahun di mana manfaat lebih besar daripada biaya (lihat Gambar 2-6). Kemudian, selisih antara NPV tahunan dan kumulatif untuk tahun tersebut dibagi dengan NPV tahunan untuk menentukan seberapa jauh titik impas akan terjadi pada tahun tersebut. Lihat Gambar 2-7 untuk perhitungan titik impas. Titik impas juga dapat digambarkan secara grafis, seperti yang ditunjukkan pada Gambar 2-8. Nilai sekarang secara kumulatif dari biaya dan manfaat untuk setiap tahun diplot pada grafik menggunakan garis; titik di mana garis bersilangan adalah titik impas.

Kalkulasi	Definisi	Formula
Present Value (PV)	Jumlah investasi saat ini dibagi dengan jumlah investasi yang sama di masa depan, dengan memperhitungkan inflasi dan waktu.	$\frac{\text{Amount}}{(1 + \text{interest rate})^n}$ n = number of years in future
Net Present Value (NPV)	Keuntungan *Present Value* (PV) dikurangi biaya *Present Value* (PV).	PV Benefits – PV Costs
Return on Investment (ROI)	Jumlah pendapatan atau penghematan biaya yang dihasilkan dari investasi tertentu.	$\frac{\text{Total benefits} - \text{Total costs}}{\text{Total costs}}$
Break-Even Point	Titik waktu di mana biaya proyek sama dengan nilai yang telah diberikan.	$\frac{\text{Yearly NPV*} - \text{Cumulative NPV}}{\text{Yearly NPV*}}$ * Gunakan jumlah NPV Tahunan dari tahun pertama di mana proyek memiliki arus kas positif. Tambahkan jumlah di atas ke tahun di mana proyek memiliki arus kas positif.

GAMBAR 2-7 Perhitungan Keuangan yang Digunakan untuk Analisis Biaya-Manfaat

Kelayakan Organisasi

Jenis terakhir dari analisis kelayakan adalah menilai kelayakan organisasi dari sistem, seberapa baik sistem akan diterima oleh penggunanya dan dimasukkan ke dalam operasi organisasi yang sedang berlangsung. Ada banyak faktor organisasi yang dapat mempengaruhi proyek, dan developer berpengalaman tahu bahwa kelayakan organisasi dapat menjadi dimensi kelayakan yang paling sulit untuk dinilai. Intinya, analisis kelayakan organisasi mencoba menjawab pertanyaan, Jika kita membangunnya, apakah mereka akan datang?

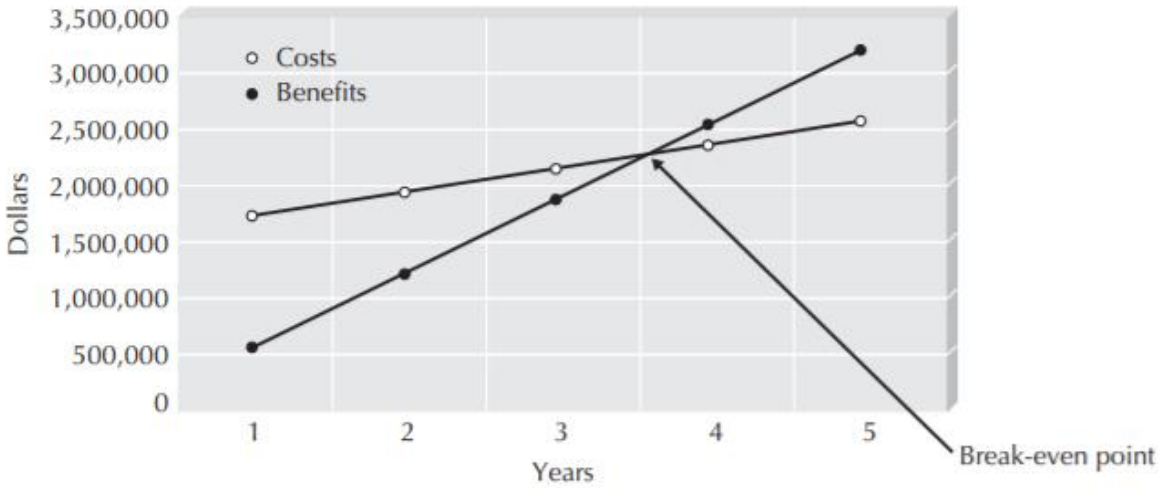
Salah satu cara untuk menilai kelayakan organisasi proyek adalah memahami seberapa baik tujuan proyek selaras dengan tujuan bisnis. Penyelarasan strategis adalah kesesuaian antara proyek dan strategi bisnis—semakin besar penyelarasan, semakin kecil risiko proyek dari perspektif kelayakan organisasi. Misalnya, jika departemen pemasaran telah memutuskan untuk lebih fokus pada pelanggan, maka proyek CRM yang menghasilkan informasi pelanggan yang terintegrasi akan memiliki keselarasan strategis yang kuat dengan tujuan pemasaran. Banyak proyek TI gagal ketika departemen TI memulainya, karena hanya tidak ada keselarasan dengan unit bisnis atau strategi organisasi.

Cara kedua untuk menilai kelayakan organisasi adalah dengan melakukan analisis pemegang kepentingan.[27] Stakeholder atau pemegang kepentingan adalah orang, kelompok, atau organisasi yang dapat mempengaruhi (atau akan dipengaruhi oleh) sistem baru. Secara umum, pemegang kepentingan yang paling penting dalam pengenalan sistem baru adalah pemenang proyek, pengguna sistem, dan manajemen organisasi (lihat Gambar 2-9), tetapi sistem terkadang juga memengaruhi pemegang kepentingan lainnya. Misalnya, departemen SI dapat menjadi pemegang kepentingan dari suatu sistem karena pekerjaan atau peran SI dapat berubah secara signifikan setelah implementasinya.

Pemenang proyek merupakan eksekutif sistem non-informasi tingkat tinggi yang biasanya merupakan sponsor proyek yang membuat permintaan sistem. Pemenang proyek mendukung proyek dengan waktu, sumber daya (misalnya, uang), dan dukungan politik dalam organisasi dengan mengkomunikasikan pentingnya sistem kepada pengambil keputusan

[27] Buku bagus yang menyajikan serangkaian teknik analisis pemegang kepentingan adalah R. O. Mason dan I. I. Mittroff, Challenging Strategic Planning Assumptions: Theory, Cases, and Techniques (New York: Wiley, 1981).

organisasi lainnya. Lebih dari satu pemenang proyek lebih disukai karena jika pemenang proyek meninggalkan organisasi, pendukungnya juga bisa pergi.



GAMBAR 2-8 Grafik Titik Impas

Sementara para pemenang proyek memberikan dukungan sehari-hari untuk sistem, dukungan manajemen organisasi meyakinkan seluruh organisasi bahwa sistem akan memberikan kontribusi yang berharga dan bahwa sumber daya yang diperlukan akan tersedia. Idealnya, manajemen harus mendorong orang-orang dalam organisasi untuk menggunakan sistem dan menerima banyak perubahan yang kemungkinan besar akan dibuat oleh sistem.

	Peran	Teknik untuk Ditingkatkan
Pemenang Proyek	Pemenang: • Memulai proyek • Mempromosikan proyek • Mengalokasikan waktu untuk proyek • Menyediakan sumber daya	• Buat presentasi tentang tujuan proyek dan manfaat yang diusulkan kepada para eksekutif yang akan memperoleh manfaat langsung dari sistem • Buat prototipe sistem untuk menunjukkan nilai potensialnya
Manajemen Organisasi	Manajer Organisasi: • Mengetahui proyek • Memberi budget uang untuk proyek • Mendorong pengguna untuk menerima dan menggunakan sistem	• Buat presentasi kepada manajemen tentang tujuan proyek dan manfaat yang diusulkan • Pasarkan manfaat sistem menggunakan memo dan buletin organisasi • Dorong pemenang proyek untuk berbicara mengenai proyek dengan rekan-rekannya

Pengguna Sistem	Pengguna: • Membuat keputusan yang memengaruhi proyek • Melakukan aktivitas langsung untuk proyek • Pada akhirnya menentukan apakah proyek berhasil, baik dengan atau tidak menggunakan sistem	• Tetapkan peran resmi pengguna di tim proyek • Tetapkan tugas khusus pengguna agar dilakukan dengan tenggat waktu yang jelas • Mintalah umpan balik reguler dari pengguna (misalnya, pada pertemuan mingguan)

GAMBAR 2-9 Beberapa Pemegang kepentingan yang Penting untuk Kelayakan Organisasi

Kelompok pemegang kepentingan penting yang ketiga adalah pengguna sistem yang pada akhirnya menggunakan sistem setelah diinstal dalam organisasi. Seringnya, tim proyek bertemu dengan pengguna di awal proyek dan kemudian menghilang setelah sistem dibuat. Dalam situasi ini, produk akhir jarang memenuhi harapan dan kebutuhan karena berubahnya kebutuhan pengguna dan pengguna menjadi lebih cerdas seiring dengan kemajuan proyek. Partisipasi pengguna harus diikutsertakan selama proses pengembangan sistem (misalnya, melakukan tugas, memberikan umpan balik, membuat keputusan).

Pada akhirnya, studi kelayakan membantu organisasi dalam membuat investasi yang lebih bijaksana dengan memaksa tim proyek untuk mempertimbangkan faktor teknis, ekonomi, dan organisasi yang dapat mempengaruhi proyek mereka. Hal ini berguna untuk melindungi tim professional TI dari kritik dengan menjaga agar unit bisnis tetap memahami keputusan dan memosisikannya sebagai pemimpin dalam proses pembuatan keputusan. Ingat, studi kelayakan harus direvisi beberapa kali selama proyek pada titik di mana tim proyek membuat keputusan penting mengenai sistem (misalnya, sebelum setiap pengulangan dari suatu proses pengembangan).

2.5 PILIHAN PROYEK

Setelah analisis kelayakan selesai, hasilnya diserahkan ke komite persetujuan, bersama dengan permintaan sistem yang telah direvisi. Komite kemudian memutuskan apakah akan menyetujui proyek, menolak proyek, atau mengajukannya sampai informasi tambahan tersedia. Di tingkat proyek, komite mempertimbangkan nilai proyek dengan memeriksa kebutuhan bisnis (dapat ditemukan dalam permintaan sistem) dan risiko dalam membangun sistem (disajikan dalam analisis kelayakan).

Namun, sebelum menyetujui proyek, komite juga mempertimbangkan proyek dari perspektif organisasi; hal itu harus diingat dalam seluruh portofolio proyek perusahaan. Cara mengelola proyek ini disebut manajemen portofolio. Manajemen portofolio mempertimbangkan berbagai jenis proyek yang ada dalam suatu organisasi—besar dan kecil, risiko tinggi dan rendah, strategis dan taktis. (Lihat Gambar 2-10 untuk mengetahui berbagai cara untuk mengklasifikasikan proyek.) Portofolio proyek yang baik memiliki campuran proyek yang tepat untuk kebutuhan organisasi. Komite bertindak sebagai manajer portofolio dengan tujuan memaksimalkan kinerja biaya-manfaat dan faktor penting lainnya dari proyek dalam portofolio mereka. Misalnya, sebuah organisasi mungkin ingin mempertahankan proyek berisiko tinggi kurang dari 20 persen dari total portofolio proyeknya.

Ukuran	Apa ukurannya? Berapa banyak orang yang dibutuhkan untuk mengerjakan proyek tersebut?
Biaya	Berapa biaya proyek yang harus ditanggung oleh organisasi?
Tujuan	Apa tujuan dari proyek tersebut? Apakah ini dimaksudkan untuk meningkatkan infrastruktur teknis? Apakah mendukung strategi bisnis saat ini? Apakah meningkatkan operasi? Apakah memberikan inovasi baru?
Lamanya	Berapa lama waktu yang dibutuhkan proyek? Berapa lama waktu yang dibutuhkan sebelum akhirnya dapat memberi nilai ke bisnis?
Risiko	Seberapa besar kemungkinan proyek tersebut akan berhasil atau gagal?
Cakupan	Seberapa besar organisasi akan dipengaruhi oleh sistem? Apakah seluruh departemen? Atau divisi tertentu? Atau seluruh korporasi?
ROI	Berapa banyak uang yang organisasi harapkan untuk diterima sebagai imbalan dari biaya proyek yang akan dikeluarkan?

GAMBAR 2-10 Cara Mengklasifikasikan Proyek

Komite persetujuan harus selektif mengenai di mana mengalokasikan sumber daya. Hal ini melibatkan trade-off di mana organisasi harus menyerahkan sesuatu sebagai imbalan untuk sesuatu yang lain demi menjaga keseimbangan portofolionya. Jika ada tiga proyek yang berpotensi memberikan hasil tinggi, namun semuanya memiliki risiko yang sangat tinggi, maka mungkin hanya satu dari proyek yang harus dipilih. Selain itu, ada kalanya sistem di tingkat proyek masuk akal secara bisnis, namun tidak masuk akal di tingkat organisasi. Dengan demikian, mungkin ada satu proyek yang dapat menunjukkan ROI tinggi dan mendukung kebutuhan bisnis yang penting bagi sebagian perusahaan, tetapi tidak dipilih. Hal ini dapat terjadi karena berbagai alasan—karena tidak adanya anggaran untuk sistem lain, karena organisasi akan mengalami beberapa jenis perubahan (misalnya, merger), karena proyek yang memenuhi persyaratan bisnis yang sama sudah berjalan, atau karena sistem tidak selaras dengan strategi perusahaan saat ini atau masa depan.

2.6 PERALATAN MANAJEMEN PROYEK TRADISIONAL

Sebelum kita benar-benar membuat rencana kerja yang sesuai untuk mengelola dan mengontrol proyek pengembangan sistem berorientasi objek, kita perlu memperkenalkan seperangkat alat manajemen proyek yang telah digunakan agar berhasil mengelola proyek pengembangan perangkat lunak tradisional (dan banyak jenis proyek lainnya): struktur rincian kerja, bagan Gantt, dan diagram jaringan. Untuk memulainya, pertama-tama kita harus memahami apa itu tugas. Tugas adalah unit kerja yang akan dilakukan oleh seorang anggota atau anggota tim pengembangan, seperti analisis kelayakan. Setiap tugas memiliki informasi seperti nama, tanggal mulai dan selesai, orang yang ditugaskan untuk menyelesaikannya, kiriman, status penyelesaian, prioritas, sumber daya yang dibutuhkan, perkiraan waktu untuk menyelesaikan tugas, dan waktu aktual yang dibutuhkan untuk menyelesaikan tugas (lihat Gambar 2-11). Hal pertama yang harus dilakukan oleh seorang manajer proyek adalah mengidentifikasi tugas-tugas yang perlu diselesaikan dan menentukan berapa lama waktu yang dibutuhkan untuk setiap tugas. Tugas dan identifikasi serta dokumentasinya adalah dasar

dari ketiga alat ini. Setelah tugas diidentifikasi dan didokumentasikan, tugas tersebut diatur dalam struktur rincian kerja yang digunakan untuk membuat bagan Gantt dan diagram jaringan yang dapat digunakan untuk menggambarkan rencana kerja tradisional secara grafis. Teknik ini membantu manajer proyek memahami dan mengelola kemajuan proyek dari waktu ke waktu.

Informasi Rencana Kerja	**Contoh**
Nama tugas	Mengerjakan kelayakan ekonomi
Tanggal mulai	5 Januari 2015
Tanggal selesai	19 Januari 2015
Orang yang ditugaskan	Sponsor proyek: Mary Smith
Kiriman	Analisis biaya-manfaat
Status pekerjaan	Terbuka
Prioritas	Tinggi
Sumber daya yang diperlukan	Software spreadsheet
Estimasi waktu	16 jam
Waktu aktual	14,5 jam

GAMBAR 2-11 Informasi Tugas

Struktur Rincian Kerja

Seorang manajer proyek dapat menggunakan pendekatan terstruktur, yaitu pendekatan atas-bawah dimana tugas tingkat tinggi didefinisikan terlebih dulu dan kemudian dipecah menjadi subtugas. Misalnya, Gambar 2-12 menunjukkan daftar tugas tingkat tinggi yang diperlukan untuk mengimplementasikan kelas pelatihan TI baru. Beberapa langkah utama dalam proses ini antara lain mengidentifikasi vendor, membuat dan mengelola survei, dan membangun ruang kelas baru. Setiap langkah kemudian dipecah secara bergantian dan diberi nomor secara hierarkis. Ada delapan subtugas (yaitu, 7.1–7.8) untuk membuat dan mengelola survei, dan ada tiga subtugas (7.2.1–7.2.3) untuk mereview tugas survei sebelumnya. Daftar tugas yang diberi nomor hierarki dengan cara ini disebut struktur rincian kerja (work breakdown structure/WBS). Jumlah tugas dan tingkat detailnya tergantung pada kompleksitas dan ukuran proyek. Minimal, WBS harus mencakup durasi tugas, status tugas saat ini (yaitu, terbuka atau selesai), dan dependensi tugas, yaitu ketika satu tugas tidak dapat dilakukan sampai tugas lain selesai. Misalnya, Gambar 2-12 menunjukkan bahwa menggabungkan perubahan pada survei (tugas 7.4) membutuhkan waktu seminggu untuk dilakukan, tetapi hal itu tidak dapat dilakukan hingga survei selesai ditinjau (tugas 7.2) dan diuji coba (tugas 7.3). Tonggak penting, atau tanggal penting, juga diidentifikasi pada rencana kerja.

Ada dua pendekatan dasar untuk mengatur WBS tradisional: dengan fase pengembangan atau produk. Misalnya, jika suatu perusahaan memutuskan bahwa perlu mengembangkan situs web, perusahaan dapat membuat WBS berdasarkan fase awal, elaborasi, konstruksi, dan transisi berdasarkan Proses Terpadu. Dalam hal ini, tugas yang akan berlangsung di awal adalah analisis kelayakan. Tugas ini akan dipecah menjadi berbagai jenis analisis kelayakan: teknis, ekonomi, dan organisasi. Masing-masing akan dipecah lebih lanjut menjadi satu set subtugas. Sebagai alternatif, perusahaan dapat mengatur rencana kerja di

sepanjang lini produk berbeda yang akan dikembangkan. Misalnya, dalam kasus situs web, produk dapat mencakup applet, server aplikasi, server database, berbagai set halaman web yang akan dirancang, peta situs, dan sebagainya. Kemudian itu semua akan didekomposisi lebih lanjut menjadi tugas-tugas berbeda yang terkait dengan fase-fase proses pengembangan. Dengan kata lain, setelah keseluruhan struktur ditentukan, tugas diidentifikasi dan dimasukkan dalam WBS. Kita akan kembali ke topik WBS dan penggunaannya dalam perencanaan berulang di bab ini nanti.

Nomor Tugas	Nama Tugas	Durasi (dalam minggu)	Dependensi	Status
1	Mengidentifikasi vendor	2		Selesai
2	Materi pelatihan peninjauan	6	1	Selesai
3	Membandingkan vendor	2	2	Dalam proses
4	Negosiasi dengan vendor	3	3	Terbuka
5	Mengembangkan informasi komunikasi	4	1	Dalam proses
6	Menyebarkan informasi	2	5	Terbuka
7	Membuat dan mengelola survei	4	6	Terbuka
7.1	Membuat surve awal	1		Terbuka
7.2	Meninjau survei awal	1	7.1	Terbuka
7.2.1	Meninjau oleh Direktur Pelatihan TI	1		Terbuka
7.2.2	Meninjau oleh Sponsor Proyek	1		Terbuka
7.2.3	Meninjau oleh Perwakilan Trainee	1		Terbuka
7.3	Uji coba survei awal	1	7.1	Terbuka
7.4	Menyertakan perubahan survei	1	7.2, 7.3	Terbuka
7.5	Membuat daftar distribusi	0.5		Terbuka

7.6	Mengirimkan survei ke daftar distribusi	0.5	7.4, 7.3	Terbuka
7.7	Mengirimkan pesan tindak lanjut	0.5	7.6	Terbuka
7.8	Mengumpulkan survei yang telah selesai	1	7.6	Terbuka
8	Menganalisa hasil dan memilih vendor	2	4, 7	Terbuka
9	Membangun kelas baru	11	1	Dalam proses
10	Mengembangkan pilihan kursus	3	8, 9	Terbuka

GAMBAR 2-12 Struktur Rincian Pekerjaan

Bagan Gantt

Bagan Gantt adalah bagan batang horizontal yang menunjukkan informasi tugas yang sama dengan WBS proyek tetapi dalam bentuk grafis. Terkadang sebuah gambar benar-benar bernilai seribu kata, dan bagan Gantt dapat mengomunikasikan status tingkat tinggi dari suatu proyek jauh lebih cepat dan lebih mudah daripada WBS. Membuat bagan Gantt itu sederhana dan dapat dilakukan menggunakan paket spreadsheet, perangkat lunak grafis, atau paket manajemen proyek project.

Pertama, tugas dibuat daftar sebagai baris dalam bagan, dan waktu dicantumkan di bagian atas secara bertahap berdasarkan kebutuhan proyek (lihat Gambar 2-13). Sebuah proyek yang pendek dapat dibagi menjadi beberapa jam atau hari, sedangkan proyek menengah dapat diwakili menggunakan minggu atau bulan. Batang horizontal digambar untuk mewakili durasi setiap tugas; awal dan akhir bilah menandai dengan tepat kapan tugas akan dimulai dan berakhir. Saat orang mengerjakan tugas, bilah yang sesuai diisi secara proporsional dengan seberapa banyak tugas yang diselesaikan. Terlalu banyak tugas pada bagan Gantt dapat membingungkan, jadi sebaiknya batasi jumlah tugas menjadi sekitar dua puluh atau tiga puluh. Jika ada lebih banyak tugas, bagi menjadi subtugas dan buat bagan Gantt untuk setiap tingkat detail.

Ada banyak hal yang dapat dilihat oleh manajer proyek dengan cepat melalui bagan Gantt. Selain melihat berapa lama tugas dan seberapa jauh prosesnya, manajer proyek juga dapat mengetahui tugas mana yang berurutan, tugas mana yang terjadi pada saat yang sama, dan tugas mana yang tumpang tindih. Dia juga bisa mendapatkan tampilan tugas yang lebih cepat dari jadwal dan tidak sesuai jadwal dengan menggambar garis vertikal pada tanggal hari tsb. Jika bilah tidak terisi dan berada di sebelah kiri garis, tugas tersebut terlambat dari jadwal.

Ada beberapa notasi khusus yang dapat ditempatkan pada bagan Gantt. Tonggak proyek ditampilkan menggunakan segitiga atau berlian terbalik. Panah digambar di antara bilah tugas untuk menunjukkan dependensi tugas. Terkadang, nama orang yang ditugaskan untuk setiap tugas dicantumkan di sebelah bilah tugas untuk menunjukkan sumber daya manusia yang telah dialokasikan untuk tugas tersebut.

Diagram Jaringan

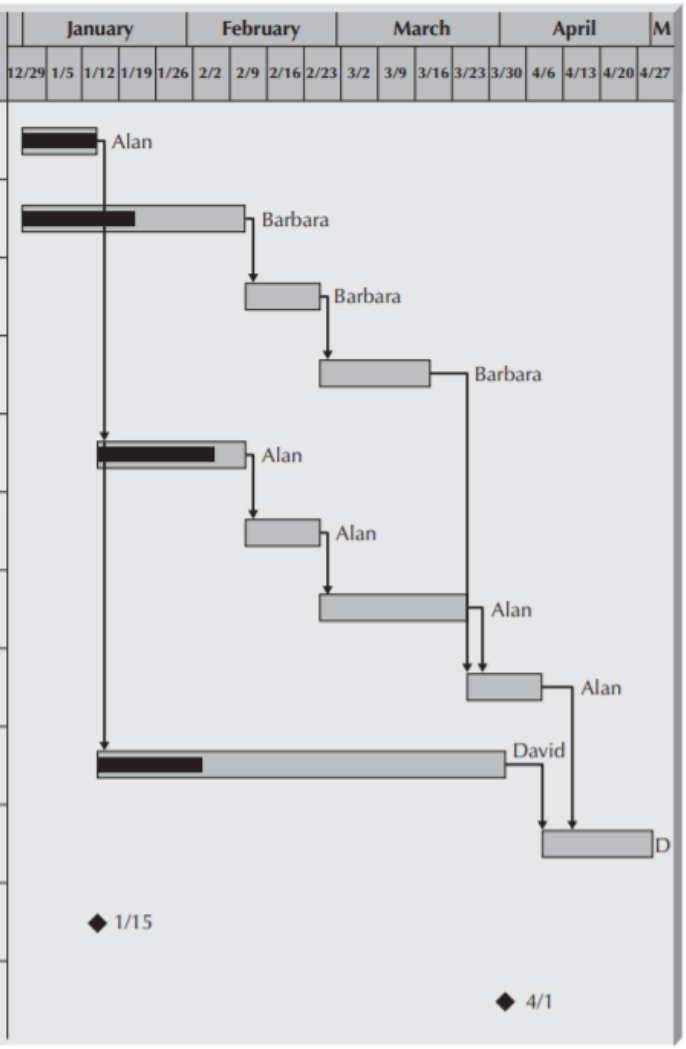
Grafis kedua untuk melihat informasi rencana kerja proyek adalah diagram jaringan yang menjabarkan tugas proyek dalam diagram alur (lihat Gambar 2-14).

Teknik Evaluasi dan Tinjauan Program (Program Evaluation and Review Technique/PERT) adalah teknik analisis jaringan yang dapat digunakan ketika perkiraan waktu tugas individu tidak pasti. Alih-alih hanya menempatkan perkiraan titik untuk memperkirakan durasi, PERT menggunakan tiga perkiraan waktu: optimis, sangat mungkin, dan pesimis. Lalu menggabungkan tiga perkiraan menjadi satu perkiraan rata-rata tertimbang menggunakan rumus berikut:

$$PERT\ rata-rata\ tertimbang = \frac{estimasi\ optimis + (4 * estimasi\ sangat\ mungkin) + estimasi\ pesimis}{6}$$

Diagram jaringan digambar sebagai jenis grafik simpul-dan-busur yang menunjukkan perkiraan waktu di simpul dan dependensi tugas pada busur. Setiap titik mewakili tugas individu, dan garis yang menghubungkan dua titik mewakili ketergantungan antara dua tugas. Tugas yang diselesaikan sebagian biasanya ditampilkan dengan garis diagonal melalui titik, dan tugas yang diselesaikan berisi garis silang.

ID	Task Name	Duration	Start	Finish	Prede
1	Identify vendors	2 wks	Wed 1/1/15	Tue 1/14/15	
2	Review training materials	6 wks	Wed 1/1/15	Tue 2/11/15	
3	Compare vendors	2 wks	Wed 2/12/15	Tue 2/25/15	2
4	Negotiate with vendors	3 wks	Wed 2/26/15	Tue 3/8/15	3
5	Develop communications information	4 wks	Wed 1/15/15	Tue 2/11/15	1
6	Disseminate information	2 wks	Wed 2/12/15	Tue 2/25/15	5
7	Create and administer survey	4 wks	Wed 2/26/15	Tue 3/25/15	6
8	Analyze results and choose	2 wks	Wed 3/26/15	Tue 4/8/15	4, 7
9	Build new classroom	11 wks	Wed 1/15/15	Tue 4/1/15	1
10	Develop course options	3 wks	Wed 4/9/15	Tue 4/29/15	8, 9
11	Budget Meeting	1 day	Wed 1/15/15	Wed 1/15/15	
12	Software Installation	1 day	Tue 4/1/15	Tue 4/1/15	



GAMBAR 2-13 Bagan Gantt

Diagram jaringan adalah cara terbaik untuk mengkomunikasikan dependensi tugas karena diagram ini menyusun tugas dalam urutan yang harus diselesaikan. Metode jalur kritis (critical path method/CPM) hanya memungkinkan identifikasi jalur kritis dalam jaringan. Jalur kritis adalah jalur terpanjang dari awal proyek sampai selesai. Jalur kritis menunjukkan semua tugas yang harus diselesaikan sesuai jadwal agar proyek secara keseluruhan selesai sesuai jadwal. Jika ada tugas di jalur kritis yang memakan waktu lebih lama dari yang diharapkan, seluruh proyek akan tertinggal. Setiap tugas di jalur kritis adalah tugas kritis, dan biasanya digambarkan dengan cara yang unik; pada Gambar 2-14 tugas itu ditunjukkan dengan batas ganda (lihat tugas 5, 6, 7, 8, dan 10). CPM dapat digunakan dengan atau tanpa PERT.

2.7 ESTIMASI UPAYA PROYEK

Ilmu (atau seni) manajemen proyek adalah dalam membuat trade-off di antara tiga konsep penting: fungsionalitas sistem, waktu untuk menyelesaikan proyek (ketika proyek akan selesai), dan biaya proyek. Pikirkan ketiga hal ini sebagai pengungkit yang saling bergantung yang dikendalikan oleh manajer proyek selama pengembangan sistem. Setiap kali satu tuas ditarik, dua tuas lainnya dipengaruhi dengan berbagai cara. Misalnya, jika seorang manajer proyek perlu menyesuaikan kembali tenggat waktu ke tanggal yang lebih awal, maka satu-satunya solusi adalah mengurangi fungsionalitas sistem atau meningkatkan biaya dengan menambahkan lebih banyak orang atau meminta mereka bekerja lembur. Seringkali, seorang manajer proyek harus bekerja dengan sponsor proyek untuk mengubah tujuan proyek, seperti mengembangkan sistem dengan fungsionalitas yang lebih sedikit atau memperpanjang tenggat waktu untuk sistem akhir, sehingga proyek memiliki tujuan yang masuk akal yang dapat dipenuhi. Di awal proyek, manajer perlu memikirkan masing-masing pengungkit ini dan kemudian terus menilai bagaimana meluncurkan proyek dengan cara yang memenuhi kebutuhan organisasi. Estimasi adalah proses menetapkan nilai yang diproyeksikan dengan tujuan waktu dan usaha. Estimasi yang dikembangkan pada awal proyek biasanya didasarkan pada kisaran nilai yang mungkin dan secara bertahap menjadi lebih spesifik seiring dengan berjalannya proyek. Artinya, kisaran nilai untuk fase awal akan jauh lebih besar daripada pada fase transisi.

Angka yang digunakan untuk menghitung perkiraan ini dapat diambil dari proyek yang memiliki tugas dan teknologi serupa atau dapat juga disediakan oleh developer berpengalaman. Secara umum, jumlahnya harus konservatif. Praktik yang baik adalah melacak nilai aktual untuk waktu dan upaya selama proses pengembangan sehingga angka dapat diperbaiki sepanjang jalan dan proyek berikutnya dapat mengambil manfaat dari data nyata tsb.

Ada berbagai cara untuk memperkirakan waktu yang dibutuhkan dalam membangun sebuah sistem. Karena Proses Terpadu disebabkan oleh Use-Case, kami menggunakan pendekatan yang didasarkan pada Use-Case: poin Use-Case.[28] Poin Use-Case, awalnya dikembangkan oleh Gustav Karner dalam Objectory AB,[29] didasarkan pada fitur unik penggunaan kasus dan orientasi objek. Dari sudut pandang praktis, untuk memperkirakan

[28] Materi di bagian ini didasarkan pada deskripsi poin kasus penggunaan yang terdapat dalam Raul R. Reed, Jr., Developing Applications with Java and UML (Reading, MA: Addison-Wesley, 2002); Geri Schneider dan Jason P. Winters, Applying Use Cases: A Practical Guide (Reading, MA: Addison-Wesley, 1998); Kirsten Ribu, "Estimating Object-Oriented Soft ware Projects with Use Cases" (Master's thesis, University of Oslo, 2001).

[29] Objectory AB diakuisisi oleh Rational pada tahun 1995 dan Rational sekarang menjadi bagian dari IBM.

upaya dalam menggunakan poin Use-Case, diagram penggunaan kasus dan Use-Case harus dibuat.[30]

Model Use-Case memiliki dua konstruksi utama: aktor dan penggunaan kasus. Aktor mewakili peran yang dimainkan oleh pengguna sistem, bukan pengguna secara spesifik. Misalnya, perannya bisa menjadi sekretaris atau manajer. Aktor juga dapat mewakili sistem lain yang akan berinteraksi dengan sistem yang sedang dikembangkan. Untuk tujuan estimasi titik Use-Case, aktor dapat diklasifikasikan menjadi sederhana, rata-rata, atau kompleks. Aktor sederhana adalah sistem terpisah yang dengannya sistem yang ada harus berkomunikasi melalui antarmuka program aplikasi (application program interface/API) yang terdefinisi dengan baik. Aktor rata-rata adalah sistem terpisah yang berinteraksi dengan sistem yang ada menggunakan protokol komunikasi standar, seperti TCP/IP, FTP, atau HTTP, atau database eksternal yang dapat diakses menggunakan SQL standar. Aktor kompleks biasanya merupakan pengguna akhir yang berkomunikasi dengan sistem. Setelah semua aktor dikategorikan ke dalam sederhana, rata-rata, atau kompleks, manajer proyek menghitung jumlah aktor di setiap kategori dan memasukkan nilai ke dalam tabel pembobotan aktor yang belum disesuaikan yang terdapat dalam lembar kerja estimasi titik Use-Case (lihat Gambar 2-15). Manajer proyek kemudian menghitung Total Bobot Aktor yang Tidak Disesuaikan (Unadjusted Actor Weight Total/UAW). Hal ini dihitung dengan menjumlahkan hasil individual yang dihitung dengan mengalikan faktor pembobotan dengan jumlah aktor dari setiap jenis yang ada. Misalnya, jika kita berasumsi bahwa diagram Use-Case memiliki nol sederhana, rata-rata nol, dan empat aktor kompleks yang berinteraksi dengan sistem yang sedang dikembangkan, UAW akan sama dengan angka 12 (lihat Gambar 2-16).

[30] Kami membahas secara detail pemodelan kasus penggunaan di bab berikutnya.

Unadjusted Actor Weighting Table:

Actor Type	Description	Weighting Factor	Number	Result
Simple	External System with well-defined API	1		
Average	External System using a protocol-based interface, e.g., HTTP, TCT/IP, or a database	2		
Complex	Human	3		
		Unadjusted Actor Weight Total (UAW)		

Unadjusted Use Case Weighting Table:

Use-Case Type	Description	Weighting Factor	Number	Result
Simple	1–3 transactions	5		
Average	4–7 transactions	10		
Complex	>7 transactions	15		
		Unadjusted Use-Case Weight Total (UUCW)		

Unadjusted Use Case Points (UUCP) = UAW + UUCW

Technical Complexity Factors:

Factor Number	Description	Weight	Assigned Value (0–5)	Weighted Value	Notes
T1	Distributed system	2.0			
T2	Response time or throughput performance objectives	1.0			
T3	End-user online efficiency	1.0			
T4	Complex internal processing	1.0			
T5	Reusability of code	1.0			
T6	Ease of installation	0.5			
T7	Ease of use	0.5			
T8	Portability	2.0			
T9	Ease of change	1.0			
T10	Concurrency	1.0			
T11	Special security objectives included	1.0			
T12	Direct access for third parties	1.0			
T13	Special user training required	1.0			
		Technical Factor Value (TFactor)			

Technical Complexity Factor (TCF) = 0.6 + (0.01 * TFactor)

Environmental Factors:

Factor Number	Description	Weight	Assigned Value (0–5)	Weighted Value	Notes
E1	Familiarity with system development process being used	1.5			
E2	Application experience	0.5			
E3	Object-oriented experience	1.0			
E4	Lead analyst capability	0.5			
E5	Motivation	1.0			
E6	Requirements stability	2.0			
E7	Part time staff	−1.0			
E8	Difficulty of programming language	−1.0			
		Environmental Factor Value (EFactor)			

Environmental Factor (EF) = 1.4 + (−0.03 * EFactor)
Adjusted Use Case Points (UCP) = UUCP * TCF * ECF
Effort in Person Hours = UCP * PHM

TEMPLATE can be found at www.wiley.com/college/dennis

FIGURE 2-15 Use-Case Point-Estimation Worksheet

GAMBAR 2-15 Lembar kerja estimasi titik Use-Case

Sebuah penggunaan kasus merepresentasikan proses bisnis utama yang akan dilakukan sistem yang menguntungkan aktor dalam berbagai cara. Penggunaan kasus dapat dikategorikan sebagai sederhana, rata-rata, atau kompleks, tergantung pada jumlah transaksi unik yang harus ditangani oleh penggunaan kasus. Sebuah penggunaan kasus diklasifikasikan sebagai sederhana jika mendukung satu hingga tiga transaksi, atau disebut rata-rata jika mendukung empat hingga tujuh transaksi, atau kompleks jika mendukung lebih dari tujuh transaksi. Setelah semua Use-Case berhasil dikategorikan, manajer proyek memasukkan

nomor setiap jenis Use-Case ke dalam tabel pembobotan Use-Case yang tidak disesuaikan yang terdapat dalam lembar kerja estimasi titik Use-Case (lihat Gambar 2-15). Dengan mengalikan dengan bobot yang sesuai dan menjumlahkan hasilnya, kita mendapatkan nilai untuk total bobot Use-Case yang tidak disesuaikan (unadjusted use-case weight total/UUCW). Misalnya, jika kita berasumsi bahwa kita memiliki tiga Use-Case sederhana, empat Use-Case rata-rata, dan satu Use-Case kompleks, nilai total bobot Use-Case yang tidak disesuaikan adalah 70 (lihat Gambar 2-16). Selanjutnya, manajer proyek menghitung nilai poin Use-Case yang tidak disesuaikan (unadjusted use-case point/UUCP) hanya dengan menjumlahkan total bobot aktor yang tidak disesuaikan dan total bobot Use-Case yang tidak disesuaikan. Dalam hal ini nilai UUCP sama dengan 82 (lihat Gambar 2-16).

Unadjusted Actor Weighting Table:

Actor Type	Description	Weighting Factor	Number	Result
Simple	External system with well-defined API	1	0	0
Average	External system using a protocol-based interface, e.g., HTTP, TCT/IP, or a database	2	0	0
Complex	Human	3	4	12
			Unadjusted Actor Weight Total (UAW)	**12**

Unadjusted Use-Case Weighting Table:

Use Case Type	Description	Weighting Factor	Number	Result
Simple	1–3 transactions	5	3	15
Average	4–7 transactions	10	4	40
Complex	>7 transactions	15	1	15
			Unadjusted Use Case Weight Total (UUCW)	**70**

Unadjusted Use-Case Points (UUCP) = UAW + UUCW **82 = 12 + 70**

Technical Complexity Factors:

Factor Number	Description	Weight	Assigned Value (0–5)	Weighted Value	Notes
T1	Distributed system	2.0	0	0	
T2	Response time or throughput performance objectives	1.0	5	5	
T3	End-user online efficiency	1.0	3	3	
T4	Complex internal processing	1.0	1	1	
T5	Reusability of code	1.0	1	1	
T6	Ease of installation	0.5	2	1	
T7	Ease of use	0.5	4	2	
T8	Portability	2.0	0	0	
T9	Ease of change	1.0	2	2	
T10	Concurrency	1.0	0	0	
T11	Special security objectives included	1.0	0	0	
T12	Direct access for third parties	1.0	0	0	
T13	Special user training required	1.0	0	0	
			Technical Factor Value (TFactor)	**15**	

Technical Complexity Factor (TCF) = 0.6 + (0.01 * TFactor) **0.75 = 0.6 + (0.01 * 15)**

Environmental Factors:

Factor Number	Description	Weight	Assigned Value (0–5)	Weighted Value	Notes
E1	Familiarity with system development process being used	1.5	4	6	
E2	Application experience	0.5	4	2	
E3	Object-oriented experience	1.0	4	4	
E4	Lead analyst capability	0.5	5	2.5	
E5	Motivation	1.0	5	5	
E6	Requirements stability	2.0	5	10	
E7	Part-time staff	−1.0	0	0	
E8	Difficulty of programming language	−1.0	4	−4.0	
			Environmental Factor Value (EFactor)	**25.5**	

Environmental Factor (EF) = 1.4 + (−0.03 * EFactor) **0.635 = 1.4 + (−0.03 * 25.5)**
Adjusted Use Case Points (UCP) = UUCP * TCF * ECF **33.3375 = 70 * 0.75 * 0.635**
Effort in person-hours = UCP * PHM **666.75 = 20 * 33.3375**

GAMBAR 2-16 Estimasi titik Use-Case untuk sistem perjanjian

Estimasi berbasis titik Use-Case juga memiliki seperangkat faktor yang digunakan untuk menyesuaikan nilai titik Use-Case. Dalam hal ini, ada dua set faktor: faktor kompleksitas teknis (technical complexity factors/TCFs) dan faktor lingkungan (environmental factors/EFs). Ada tiga belas faktor teknis yang terpisah dan delapan faktor lingkungan yang terpisah. Tujuan dari faktor-faktor ini adalah untuk memungkinkan proyek secara keseluruhan dievaluasi mengenai kompleksitas sistem yang dikembangkan dan tingkat pengalaman staf pengembangan. Jelas, jenis faktor ini dapat memengaruhi upaya yang diperlukan tim untuk mengembangkan sistem. Masing-masing faktor ini diberi nilai antara 0 dan 5, 0 menunjukkan bahwa faktor tersebut tidak relevan dengan sistem yang sedang dipertimbangkan dan 5 menunjukkan bahwa faktor tersebut penting agar sistem berhasil. Nilai yang diberikan kemudian dikalikan dengan bobotnya masing-masing. Nilai bobot ini kemudian dijumlahkan untuk menghasilkan nilai faktor teknis (technical factor value/TFactor) dan nilai faktor lingkungan (environmental factor value/EFactor) (lihat Gambar 2-15).

Faktor teknis meliputi hal-hal berikut ini (lihat Gambar 2-15):

- Apakah sistem akan menjadi sistem terdistribusi
- Pentingnya waktu respon
- Tingkat efisiensi pengguna akhir yang menggunakan sistem
- Kompleksitas pemrosesan internal sistem
- Pentingnya penggunaan kembali kode
- Seberapa mudah proses instalasinya
- Pentingnya kemudahan dalam menggunakan sistem
- Seberapa penting bagi sistem untuk dapat di-porting ke platform lain
- Apakah pemeliharaan sistem itu penting
- Apakah sistem harus menangani pengolahan paralel dan bersamaan
- Tingkat keamanan khusus yang diperlukan
- Tingkat akses sistem oleh pihak ketiga
- Apakah pelatihan pengguna akhir khusus diperlukan.

Dengan asumsi nilai faktor teknis adalah T1 (0), T2 (5), T3 (3), T4 (1), T5 (1), T6 (2), T7 (4), T8 (0), T9 (2), T10 (0), T11 (0), T12 (0), dan T13 (0), nilai faktor teknis (technical factor value/TFactor) dihitung sebagai jumlah tertimbang dari faktor teknis individu. Dalam hal ini TFactor sama dengan 15 (lihat Gambar 2-16). Memasukkan nilai ini ke dalam persamaan faktor kompleksitas teknis (TCF) (0,6 ± 1 (0,01 * TFactor)) dari lembar kerja titik Use-Case memberikan nilai 0,75 untuk TCF sistem (lihat Gambar 2-15 dan 2- 16).

Faktor lingkungan meliputi hal-hal berikut ini (lihat Gambar 2-15):

- Tingkat pengalaman yang dimiliki staf pengembangan dengan proses pengembangan yang sedang digunakan
- Aplikasi yang sedang dikembangkan
- Tingkat pengalaman dalam berorientasi objek
- Tingkat kemampuan analis utama
- Tingkat motivasi tim pengembangan untuk mengirimkan sistem
- Stabilitas persyaratan
- Apakah staf paruh waktu harus dimasukkan sebagai bagian dari tim pengembangan
- Kesulitan bahasa pemrograman yang digunakan untuk mengimplementasikan sistem

Dengan asumsi nilai faktor lingkungan E1 (4), E2 (4), E3 (4), E4 (5), E5 (5), E6 (5), E7 (0), dan E8 (4) memberikan nilai faktor lingkungan (environmental factor value/EFactor) sebesar 25,5 (lihat Gambar 2-16). Seperti TFactor, Efactor hanyalah jumlah dari nilai-nilai

terbobot. Menggunakan persamaan faktor lingkungan (EF) (1,4 ± 1 (-0,03 * EFactor)) dari lembar kerja titik Use-Case menghasilkan nilai 0,635 untuk EF sistem (lihat Gambar 2-15 dan 2-16). Memasukkan nilai TCF dan EF, bersama dengan nilai UUCP yang dihitung sebelumnya, ke dalam persamaan poin Use-Case yang disesuaikan (UUCP * TCF * EF) dari lembar kerja menghasilkan nilai 33,3375 titik Use-Case yang disesuaikan (UCP) (lihat Gambar 2-16).

Sekarang, setelah kita mengetahui perkiraan ukuran sistem melalui nilai poin Use-Case yang disesuaikan, kita siap untuk memperkirakan upaya yang diperlukan untuk membangun sistem. Dalam karya asli Karner, dia menyarankan untuk mengalikan jumlah poin Use-Case dengan 20 untuk memperkirakan jumlah jam kerja yang dibutuhkan dalam membangun sistem. Namun, berdasarkan pengalaman tambahan dalam menggunakan poin Use-Case, aturan keputusan untuk menentukan nilai pengganda orang-jam (person-hours multiplier/PHM) telah dibuat yang menyarankan penggunaan 20 atau 28, berdasarkan nilai yang ditetapkan untuk faktor lingkungan individu. Aturan keputusannya adalah:

Jika jumlah dari (total nilai yang ditetapkan Efactors E1 sampai E6 < 3) dan (jumlah nilai yang ditetapkan Efactors E7 dan E8 > 3) ≤ 2

PHM = 20

Selain itu, jika jumlah dari (total nilai yang ditetapkan Efactors E1 sampai E6 < 3) dan (jumlah nilai yang ditetapkan Efactors E7 dan E8 > 3) = 3 atau 4

PHM = 28

Selain itu, pikirkan kembali mengenai proyek; proyek itu memiliki risiko gagal yang sangat tinggi.

Berdasarkan aturan tersebut, karena tidak ada Efactors E1 sampai E6 yang memiliki nilai kurang dari 3 dan hanya Efactor E8 yang memiliki nilai lebih besar dari 3, maka jumlah EFactors-nya adalah 1. Oleh karena itu, sistem harus menggunakan PHM sebesar 20. Memasukkan nilai UCP (33,3375) dan PHM (20) ke dalam persamaan usaha (UCP * PHM) memberikan perkiraan jumlah jam kerja 666,75 jam (lihat Gambar 2-15 dan 2-16).

2.8 MEMBUAT DAN MENGELOLA RENCANA KERJA

Setelah manajer proyek memiliki gambaran umum tentang fungsionalitas dan upaya untuk proyek tersebut, dia membuat rencana kerja, yang merupakan jadwal dinamis yang mencatat dan melacak semua tugas yang perlu diselesaikan selama proyek berlangsung. Rencana kerja mencantumkan setiap tugas, bersama dengan informasi penting, seperti kapan tugas itu harus diselesaikan, orang yang ditugaskan untuk melakukan pekerjaan itu, dan apa yang akan dihasilkan. Tingkat detail dan jumlah informasi yang ditangkap oleh rencana kerja bergantung pada kebutuhan proyek, dan detailnya biasanya meningkat seiring dengan berjalannya proyek.

Tujuan keseluruhan untuk sistem harus dicantumkan pada permintaan sistem, dan merupakan tugas manajer proyek untuk mengidentifikasi semua tugas yang perlu diselesaikan untuk memenuhi tujuan tersebut. Hal ini terdengar seperti tugas yang menakutkan. Bagaimana seseorang bisa mengetahui segala sesuatu yang perlu dilakukan untuk membangun sistem yang belum pernah dibangun sebelumnya?

Salah satu pendekatan untuk mengidentifikasi tugas adalah mendapatkan daftar tugas yang telah dikembangkan dan memodifikasinya. Ada daftar tugas standar, atau metodologi, yang tersedia untuk digunakan sebagai titik awal. Seperti yang kami nyatakan di bab

sebelumnya, metodologi adalah pendekatan formal untuk menerapkan proses pengembangan sistem (yaitu, daftar langkah dan hasil). Manajer proyek dapat mengambil metodologi yang ada, memilih langkah dan hasil yang berlaku untuk proyek yang sedang dikerjakan, dan menambahkannya ke rencana kerja. Jika metodologi yang ada tidak tersedia dalam organisasi, metodologi dapat dibeli dari konsultan atau vendor, atau buku seperti buku teks ini dapat berfungsi sebagai panduan. Karena sebagian besar organisasi memiliki metodologi yang mereka gunakan untuk proyek, menggunakan metodologi yang ada adalah cara paling populer untuk membuat rencana kerja. Dalam kasus kami, karena kami menggunakan metodologi berbasis Proses Terpadu, kami dapat menggunakan fase, alur kerja, dan pengulangan sebagai titik awal untuk membuat struktur rincian kerja evolusioner dan rencana kerja berulang.

Struktur Rincian Kerja Evolusioner dan Rencana Kerja Berulang[31]

Karena pendekatan sistem berorientasi objek untuk analisis dan desain sistem mendukung pengembangan bertahap dan berulang, setiap pendekatan perencanaan proyek untuk pengembangan sistem berorientasi objek juga memerlukan proses bertahap dan berulang. Dalam deskripsi Proses Terpadu yang ditingkatkan di bab sebelumnya, proses pengembangan diatur di sekitar pengulangan, fase, dan alur kerja. Dalam banyak hal, rencana kerja untuk proses pengembangan bertahap dan berulang diatur dengan cara yang sama. Untuk setiap pengulangan, ada tugas berbeda yang dieksekusi pada setiap alur kerja. Bagian ini menjelaskan proses bertahap dan berulang menggunakan WBS evolusioner untuk perencanaan proyek yang dapat digunakan dengan pengembangan sistem berorientasi objek.

WBS evolusioner memungkinkan analis untuk mengembangkan rencana kerja berulang. Pertama, WBS evolusioner diatur secara standar di semua proyek: dengan alur kerja, fase, dan kemudian tugas spesifik yang diselesaikan selama pengulangan individu. Kedua, WBS evolusioner dibuat secara bertahap dan berulang. Hal ini memberi pandangan yang lebih realistis mengenai estimasi biaya dan jadwal. Ketiga, karena struktur WBS evolusioner tidak terikat pada proyek tertentu, WBS evolusioner memungkinkan perbandingan proyek yang dikerjakan saat ini dengan proyek sebelumnya. Dari sini kita dapat belajar dari keberhasilan dan kegagalan masa lalu.

Dalam hal Proses Terpadu yang ditingkatkan, alur kerja adalah poin utama yang tercantum dalam WBS. Selanjutnya, setiap alur kerja didekomposisi sepanjang fase Proses Terpadu yang ditingkatkan. Setelah itu, setiap fase didekomposisi sepanjang tugas yang harus diselesaikan untuk membuat kiriman yang terkait dengan pengulangan individu yang terdapat dalam setiap fase (lihat Gambar 1-16). Template dua tingkat pertama dari WBS evolusioner untuk Proses Terpadu yang ditingkatkan akan terlihat seperti Gambar 2-17.

Saat setiap pengulangan yang melalui proses pengembangan selesai, pengulangan dan tugas tambahan ditambahkan ke WBS (misalnya, WBS berkembang bersama dengan sistem informasi yang berkembang). [32] Misalnya, aktivitas khas untuk fase awal alur kerja manajemen proyek antara lain mengidentifikasi proyek, melakukan analisis kelayakan, memilih proyek, dan mengestimasi upaya. Fase awal dari alur kerja persyaratan akan mencakup penentuan pengumpulan persyaratan dan teknik analisis, mengidentifikasi persyaratan fungsional dan nonfungsional, mewawancarai pemegang kepentingan, mengembangkan dokumen visi, dan mengembangkan Use-Case. Mungkin tidak ada tugas yang berhubungan dengan fase awal operasi dan yang mendukung alur kerja. Contoh WBS

[31] Materi dalam di bagian ini didasarkan pada Walker Royce, Software Project Management: A Unified Framework (Reading, MA: Addison-Wesley, 1998).

[32] Sumber bagus yang dapat membantu menjelaskan pendekatan ini adalah Phillippe Krutchen, "Planning an Iterative Project," The Rational Edge (Oktober 2002); Eric Lopes Cordoza dan D. J. de Villiers, "Project Planning Best Practices," The Rational Edge (Agustus 2003)

evolusioner untuk merencanakan fase awal Proses Terpadu yang ditingkatkan, berdasarkan Gambar 1-16 dan 2-17, ditunjukkan pada Gambar 2-18. Perhatikan bahwa dua tugas terakhir untuk alur kerja manajemen proyek adalah "membuat rencana kerja untuk pengulangan pertama fase elaborasi" dan "menilai fase awal"; dua hal terakhir yang harus dilakukan adalah merencanakan pengulangan berikutnya dalam pengembangan sistem yang berkembang dan menilai pengulangan yang dilakukan saat ini. Saat proyek bergerak melalui fase selanjutnya, setiap alur kerja memiliki tugas yang ditambahkan ke pengulangannya. Misalnya, alur kerja analisis akan memiliki model fungsional, struktural, dan perilaku selama fase elaborasi. Pada akhirnya, ketika sebuah pengulangan mencakup banyak tugas kompleks, kita dapat menggunakan peralatan tradisional, seperti bagan Gantt dan diagram jaringan, yang dapat digunakan untuk merinci rencana kerja untuk pengulangan spesifik tersebut.

I. Pemodelan Bisnis a. Permulaan b. Elaborasi c. Konstruksi d. Transisi e. Produksi II. Persyaratan a. Permulaan b. Elaborasi c. Konstruksi d. Transisi e. Produksi III. Analisis a. Permulaan b. Elaborasi c. Konstruksi d. Transisi e. Produksi IV. Desain a. Permulaan b. Elaborasi c. Konstruksi d. Transisi e. Produksi	V. Implementasi a. Permulaan b. Elaborasi c. Konstruksi d. Transisi e. Produksi VI. Pengujian a. Permulaan b. Elaborasi c. Konstruksi d. Transisi e. Produksi VII. Penyebaran a. Permulaan b. Elaborasi c. Konstruksi d. Transisi e. Produksi VIII. Konfigurasi dan Manajemen Perubahan a. Permulaan b. Elaborasi c. Konstruksi d. Transisi e. Produksi	IX. Manajemen Proyek a. Permulaan b. Elaborasi c. Konstruksi d. Transisi e. Produksi X. Lingkungan a. Permulaan b. Elaborasi c. Konstruksi d. Transisi e. Produksi XI. Operasi dan Dukungan a. Permulaan b. Elaborasi c. Konstruksi d. Transisi e. Produksi XII. Manajemen Infrastruktur a. Permulaan b. Elaborasi c. Konstruksi d. Transisi e. Produksi

GAMBAR 2-17 Template WBS Evolusioner untuk Proses Terpadu yang Ditingkatkan

	Durasi	Dependensi
I. Pemodelan Bisnis a. Permulaan 1. Memahami situasi bisnis saat ini 2. Mengungkap masalah proses bisnis 3. Mengidentifikasi proyek potensial b. Elaborasi	 0.5 hari 0.25 hari	

	c. Konstruksi d. Transisi e. Produksi	0.25 hari	
II.	Persyaratan a. Permulaan 1. Mengidentifikasi teknik analisis persyaratan yang tepat 2. Mengidentifikasi teknik pengumpulan persyaratan yang tepat 3. Mengidentifikasi persyaratan fungsional dan nonfungsional A. Melakukan sesi JAD B. Melakukan analisis dokumen C. Melakukan wawancara 1. Mewawancara sponsor proyek 2. Mewawancara kontak sistem inventaris 3. Mewawancara kontak sistem order khusus 4. Mewawancara kontak ISP 5. Mewawancara kontak web pilihan CD 6. Mewawancara personel lain D. Mengamati proses toko ritel 4. Menganalisis sistem yang sudah ada 5. Membuat definisi persyaratan A. Menentukan persyaratan untuk dilacak B. Gabungkan persyaratan jika diminta C. Meninjau persyaratan dengan sponsor b. Elaborasi c. Konstruksi d. Transisi e. Produksi	0.25 hari 0.25 hari 3 hari 5 hari 0.5 hari 0.5 hari 0.5 hari 0.5 hari 0.5 hari 1 hari 0.5 hari 4 hari	II.a.1, II.a.2 II.a.3.A II.a.3.A II.a.3.A II.a.1, II.a.2 II.a.3, II.a.4

	1 hari 5 hari 2 hari	 II.a.5.A II.a.5.B
III. Analisis a. Permulaan 1. Mengidentifikasi proses bisnis 2. Mengidentifikasi penggunaan kasus b. Elaborasi c. Konstruksi d. Transisi e. Produksi	 3 hari 3 hari	 III.a.1
IV. Desain a. Permulaan 1. Mengidentifikasi kelas potensial b. Elaborasi c. Konstruksi d. Transisi e. Produksi	 3 hari	 III.a
V. Implementasi a. Permulaan b. Elaborasi c. Konstruksi d. Transisi e. Produksi		
VI. Pengujian a. Permulaan b. Elaborasi c. Konstruksi d. Transisi e. Produksi		
VII. Penyebaran a. Permulaan b. Elaborasi c. Konstruksi d. Transisi e. Produksi		
VIII. Konfigurasi dan Manajemen Perubahan a. Permulaan		

	1. Mengidentifikasi kontrol akses yang diperlukan untuk artefak yang dikembangkan 2. Mengidentifikasi mekanisme kontrol versi untuk artefak yang dikembangkan b. Elaborasi c. Konstruksi d. Transisi e. Produksi	0.25 hari 0.25 hari	
IX.	Manajemen Proyek a. Permulaan 1. Membuat rencana kerja untuk fase awal 2. Membuat permintaan sistem 3. Melakukan analisis kelayakan A. Melakukan analisis kelayakan teknis B. Melakukan analisis kelayakan ekonomi C. Melakukan analisis kelayakan organisasi 4. Mengidentifikasi upaya proyek 5. Mengidentifikasi persyaratan staf 6. Menghitung perkiraan biaya 7. Membuat rencana kerja untuk pengulangan pertama fase elaborasi 8. Menilai fase permulaan b. Elaborasi c. Konstruksi d. Transisi e. Produksi	 1 hari 1 hari 1 hari 2 hari 2 hari 0.5 hari 0.5 hari 0.5 hari 1 hari 1 hari	 IX.a.2 IX.a.3 IX.a.4 IX.a.5 IX.a.1 I.a, II.a, III.a, IV.a, V.a, VI.a, VII.a, VIII.a, IX.a, X.a, XI.a, XII.a
X.	Lingkungan a. Permulaan 1. Memperoleh dan menginstal CASE tool		

	2. Memperoleh dan menginstal pemrograman lingkungan 3. Memperoleh dan menginstal peralatan konfigurasi dan manajemen perubahan 4. Memperoleh dan menginstal peralatan manajemen proyek b. Elaborasi c. Konstruksi d. Transisi e. Produksi	0.25 hari 0.25 hari 0.25 hari 0.25 hari	
XI.	Operasi dan Dukungan a. Permulaan b. Elaborasi c. Konstruksi d. Transisi e. Produksi		
XII.	Manajemen Infrastruktur a. Permulaan 1. Mengidentifikasi standar dan model perusahaan yang sesuai 2. Mengidentifikasi peluang penggunaan kembali, seperti pola, kerangka kerja, dan perpustakaan 3. Mengidentifikasi proyek lama yang mirip b. Elaborasi c. Konstruksi d. Transisi e. Produksi	0.25 hari 0.5 hari 0.25 hari	

GAMBAR 2-18 WBS Evolusioner untuk Fase Permulaan Berbasis Pengulangan Tunggal

Lingkup Pengelolaan

Seorang analis dapat berasumsi bahwa sebuah proyek akan aman dari masalah penjadwalan karena dia telah hati-hati dalam memperkirakan dan merencanakan proyek di awal. Namun, alasan paling umum mengenai jadwal dan pembengkakan biaya—penyelewengan lingkup (scope creep)—terjadi setelah proyek berjalan. Penyelewengan

lingkup terjadi ketika persyaratan baru ditambahkan ke proyek setelah lingkup proyek asli didefinisikan dan dibekukan. Hal ini dapat terjadi karena berbagai alasan: Pengguna mungkin tiba-tiba memahami potensi sistem baru dan menyadari fungsionalitas baru yang akan berguna; developer mungkin menemukan kemampuan menarik; seorang manajer senior mungkin memutuskan untuk membiarkan sistem ini mendukung strategi baru yang dikembangkan pada rapat dewan sebelumnya.

Untungnya, dengan menggunakan proses pengembangan berulang dan bertahap memungkinkan tim untuk menangani perubahan persyaratan dengan cara yang efektif. Namun, semakin luas perubahannya, semakin besar dampaknya terhadap biaya dan jadwal. Utamanya adalah mengidentifikasi kebutuhan sebaik mungkin di awal proyek dan menerapkan teknik analisis secara efektif. Misalnya, jika kebutuhan tidak jelas pada awal proyek, kombinasi pertemuan intensif dengan pengguna dan pembuatan prototipe akan memungkinkan pengguna untuk "mengalami" persyaratan dan memvisualisasikan dengan lebih baik mengenai bagaimana sistem dapat mendukung kebutuhan mereka.

Tentu saja, beberapa persyaratan mungkin terlewatkan, meskipun sudah mengambil tindakan pencegahan apapun. Namun, manajer proyek hanya boleh mengizinkan persyaratan tambahan yang benar-benar diperlukan setelah proyek mulai dijalankan. Bahkan pada saat itu, anggota tim proyek harus hati-hati dalam menilai konsekuensi dari penambahan dan keberadaan penilaian pengguna. Setiap perubahan yang diterapkan harus dilacak dengan cermat sehingga menghasilkan jejak audit yang akan berguna untuk mengukur dampak perubahan tersebut.

Terkadang perubahan tidak dapat dimasukkan ke dalam sistem yang sedang berjalan saat ini meskipun perubahan itu benar-benar akan bermanfaat. Dalam hal ini, penambahan ini harus dicatat sebagai penyempurnaan sistem di masa mendatang. Manajer proyek dapat menawarkan untuk menyediakan fungsionalitas dalam rilis sistem yang akan datang, sehingga dapat mengatakan "tidak" pada seseorang di masa mendatang.

Beberapa teknik agile yang berguna untuk mengelola ruang lingkup proyek sambil memuaskan klien adalah rapat scrum harian dan jaminan simpanan produk yang digunakan dengan Scrum. Pada dasarnya pertemuan scrum harian adalah pertemuan yang sangat singkat, biasanya lima belas menit, yang berguna bagi tim developer untuk selalu mengetahui status terkini dari sistem yang berkembang. Isi pertemuan biasanya hanya mencakup apa yang telah dicapai sejak pertemuan sebelumnya, apa yang akan dicapai sebelum pertemuan berikutnya, dan hambatan apa yang bisa muncul yang dapat menghambat kemajuan. Selain itu, fitur-fitur baru yang diminta dapat diajukan. Namun, semua fitur tambahan yang diusulkan hanya ditambahkan ke backlog produk yang dapat dipertimbangkan selama pengulangan atau timebox berikutnya (sprint dalam nomenklatur Scrum). Produk backlog pada dasarnya adalah daftar prioritas kebutuhan fungsional yang akan diselesaikan selama pengulangan. Di Scrum, hanya klien yang diizinkan untuk mengubah backlog produk. Dengan cara ini, tim pengembangan selalu memiliki daftar kumpulan persyaratan kritis. Selama proyeknya relatif kecil, pendekatan manajemen ruang lingkup ini sangat efektif.

Timeboxing

Pendekatan lain untuk manajemen ruang lingkup adalah suatu teknik yang bernama timeboxing. Sampai pada titik ini, kami telah menjelaskan mengenai proyek berorientasi tugas. Dengan kata lain, kami telah menggambarkan proyek yang memiliki jadwal, yang didasari oleh tugas yang harus diselesaikan, sehingga semakin banyak tugas dan persyaratan, semakin lama proyeknya. Beberapa perusahaan tidak begitu sabar terhadap royek pengembangan yang memakan waktu lama, dan perusahaan-perusahaan ini mengambil pendekatan berorientasi waktu yang meletakkan deadline di atas fungsionalitas.

Pikirkan tentang penggunaan perangkat lunak pengolah kata. Dari 80 persen waktunya, hanya 20 persen untuk fitur, seperti pemeriksa ejaan, huruf tebal, serta cut dan paste, yang digunakan. Fitur lain, seperti penggabungan dokumen dan pembuatan label surat, mungkin bagus, tetapi itu bukan bagian dari kebutuhan sehari-hari. Hal yang sama berlaku untuk aplikasi perangkat lunak lainnya; sebagian besar pengguna hanya mengandalkan sebagian kecil dari kemampuan perangkat lunak tsb. Ironisnya, sebagian besar developer setuju bahwa biasanya 75 persen dari keseluruhan sistem dapat dibuat dengan relatif cepat, sementara 25 persen fungsi lainnya menuntut tergantung pada lamanya waktu.

Untuk mengatasi ketidaksesuaian ini, teknik timeboxing menjadi populer, terutama ketika menggunakan metodologi RAD dan agile. Teknik ini menetapkan tenggat waktu yang tetap untuk sebuah proyek dan menyelesaikan sistem pada tenggat waktu itu, entah apa pun yang terjadi, bahkan jika perlu, fungsionalitasnya dapat dikurangi. Timeboxing memastikan bahwa tim proyek tidak terpaku pada sentuhan akhir yang dapat mengakibatkan pada molornya proyek, dan hal itu yang penting dapat memuaskan bisnis dengan menyelesaikan produk dalam jangka waktu yang relatif singkat.

Ada beberapa langkah dalam mengimplementasikan timeboxing pada sebuah proyek. Pertama, tetapkan tanggal pengiriman untuk tujuan yang diusulkan. Tenggat waktu mestinya tidak mustahil untuk dipenuhi, sehingga cara yang terbaik adalah dengan membiarkan tim proyek menentukan sendiri tanggal jatuh tempo yang realistis. Jika Anda ingat di bab sebelumnya, metodologi agile Scrum menetapkan semua kotak waktunya (sprint) menjadi tiga puluh hari kerja. Selanjutnya, membangun inti dari sistem yang akan disampaikan; Anda akan menemukan bahwa timeboxing membantu menciptakan rasa urgensi dan membantu menjaga fokus pada fitur yang paling penting. Karena jadwalnya benar-benar tetap, fungsionalitas yang tidak dapat diselesaikan dapat ditunda. Akan membantu apabila tim memprioritaskan daftar fitur sebelumnya untuk melacak fungsionalitas yang benar-benar dibutuhkan pengguna. Kualitas tidak dapat dikompromikan, terlepas dari kendala lainnya, jadi penting bahwa waktu yang dialokasikan untuk aktivitas tidak dipersingkat kecuali persyaratan diubah (misalnya, jangan mengurangi waktu yang dialokasikan untuk pengujian tanpa mengurangi fitur). Pada akhir periode waktu, sistem berkualitas tinggi diselesaikan, tetapi kemungkinan pengulangan di masa mendatang akan diperlukan untuk membuat perubahan dan penyempurnaan. Dalam hal ini, pendekatan timeboxing dapat digunakan sekali lagi.

Menyempurnakan Estimasi

Estimasi yang dihasilkan selama tahap permulaan perlu disempurnakan seiring berjalannya proyek. Hal ini bukan berarti bahwa estimasi dilakukan dengan buruk pada awal proyek; sebaliknya, hampir tidak mungkin untuk mengembangkan penilaian yang tepat dari jadwal proyek pada awal proses pengembangan. Seorang manajer proyek harus berekspektasi untuk puas dengan berbagai perkiraan yang menjadi lebih spesifik karena produk proyek dapat didefinisikan menjadi lebih baik.

Selama perencanaan, ketika sebuah sistem pertama kali diminta, sponsor proyek dan manajer proyek mencoba untuk memprediksi berapa lama proses pengembangan akan berlangsung, berapa biayanya, dan apa yang akhirnya dapat dilakukan oleh produk itu ketika dikirimkan (yaitu, fungsinya). Namun, estimasi didasarkan pada pengetahuan yang sangat sedikit mengenai sistem. Saat sistem bergerak ke elaborasi, lebih banyak informasi dikumpulkan, konsep sistem dikembangkan, dan estimasi menjadi lebih akurat dan tepat. Saat sistem bergerak mendekati penyelesaian, akurasi dan presisi meningkat, hingga akhirnya produknya dikirimkan.

Menurut salah satu ahli terkemuka dalam pengembangan perangkat lunak, rencana proyek yang dilakukan dengan baik (disiapkan pada akhir tahap permulaan) memiliki margin

kesalahan 100 persen untuk biaya proyek dan margin kesalahan 25 persen untuk jadwal. Dengan kata lain, jika rencana proyek yang dilakukan dengan hati-hati memperkirakan bahwa sebuah proyek akan menelan biaya Rp. 1.400.000.000 dan memakan waktu dua puluh minggu, proyek tersebut sebenarnya akan menelan biaya antara Rp. 0 dan Rp. 2.800.000.000 dan memakan waktu antara lima belas hingga dua puluh lima minggu.

Apa yang terjadi jika Anda melampaui estimasi (misalnya, analisis berakhir dua minggu lebih lama dari yang diharapkan)? Ada beberapa cara untuk menyesuaikan estimasi di masa mendatang. Jika tim proyek menyelesaikan selangkah lebih cepat dari jadwal, sebagian besar manajer proyek menggeser tenggat waktu lebih cepat dengan jumlah yang sama tetapi tidak menyesuaikan tanggal penyelesaian yang dijanjikan. Meskipun begitu, tantangannya terjadi ketika tim proyek terlambat memenuhi tanggal yang dijadwalkan. Tiga kemungkinan tanggapan terhadap tanggal jadwal yang terlewat disajikan pada Gambar 2-19. Jika di awal proyek estimasi terbukti terlalu optimis, perencana tidak boleh berharap untuk menebus waktu yang hilang—sangat sedikit proyek yang akhirnya melakukan hal ini. Sebaliknya, mereka harus mengubah perkiraan masa mendatang untuk memasukkan peningkatan yang serupa dengan yang dialami. Misalnya, jika fase pertama diselesaikan lebih dari 10 persen dari jadwal, perencana harus meningkatkan sisa perkiraan mereka sebesar 10 persen.

Asumsi	**Aksi**	**Tingkat Risiko**
Jika Anda menganggap sisa proyeknya lebih sederhana dibanding bagian yang terlambat dan juga lebih sederhana dibanding dari yang diyakini saat perkiraan jadwal asli dibuat, Anda dapat mengganti waktu yang hilang.	Jangan mengganti jadwal	Risiko tinggi
Jika Anda menganggap sisa proyek lebih sederhana dibanding bagian yang terlambat dan tidak lebih kompleks dari perkiraan semula, Anda tidak dapat mengganti waktu yang hilang, tetapi Anda tidak akan kehilangan waktu untuk mengerjakan sisa proyek.	Tingkatkan seluruh jadwal dengan menjumlahkan waktu yang Anda lewati (misalnya, jika Anda melewatkan tanggal yang dijadwalkan dua minggu, pindahkan sisa tanggal jadwal ke dua minggu kemudian). Jika Anda memasukkan waktu tambahan di akhir proyek dalam jadwal asli, Anda mungkin tidak perlu mengubah tanggal pengiriman sistem yang dijanjikan; Anda hanya akan menghabiskan waktu santai.	Risiko sedang
Jika Anda berasumsi bahwa sisa proyek sama rumitnya dengan bagian yang terlambat (perkiraan awal Anda terlalu optimis), maka semua tanggal yang dijadwalkan di masa mendatang menyepelekan	Tingkatkan seluruh jadwal dengan persentase minggu yang Anda lewati (misalnya, jika Anda terlambat dua minggu pada bagian proyek yang seharusnya memakan	Risiko rendah

waktu nyata yang dibutuhkan dengan persentase yang sama dengan bagian yang terlambat.	waktu delapan minggu, Anda perlu meningkatkan semua perkiraan waktu yang tersisa sebesar 25 persen). Jika hal ini menyebabkan pemindahan tanggal pengiriman baru melampaui apa yang dapat diterima oleh sponsor proyek, ruang lingkup proyek harus dikurangi.	

GAMBAR 2-19 Kemungkinan Tindakan Ketika Tanggal Jadwal Terlambat

Mengelola Risiko

Salah satu aspek terakhir dari manajemen proyek adalah manajemen risiko, yaitu proses penilaian dan penanganan risiko yang terkait dengan pengembangan proyek. Banyak hal yang dapat menyebabkan risiko, seperti personel yang lemah, scope creep, desain yang buruk, dan estimasi yang terlalu optimis. Tim proyek harus menyadari potensi risiko sehingga masalah dapat dihindari atau dikendalikan dengan baik sebelumnya.

Biasanya, tim proyek membuat penilaian risiko, atau dokumen yang melacak potensi risiko bersama dengan evaluasi kemungkinan setiap risiko dan potensi dampaknya terhadap proyek (Gambar 2-20). Satu atau dua paragraf juga disertakan untuk menjelaskan cara-cara potensial untuk menangani risiko tsb. Ada banyak pilihan: Risiko dapat dipublikasikan, dihindari, atau bahkan dihilangkan dengan mengatasi akar masalahnya. Misalnya, bayangkan tim proyek berencana menggunakan teknologi baru tetapi anggotanya mengidentifikasi risiko bahwa mereka tidak memiliki keterampilan teknis yang tepat. Mereka percaya bahwa tugas mungkin akan memakan waktu lebih lama untuk dilakukan karena kurva belajar yang tinggi. Salah satu rencananya adalah menghilangkan akar penyebab risiko—kurangnya pengalaman teknis oleh anggota tim—dengan mencari waktu dan sumber daya yang dibutuhkan untuk memberikan pelatihan yang tepat kepada tim.

Sebagian besar manajer proyek mengikuti perkembangan potensi risiko, bahkan memprioritaskannya sesuai dengan besarnya dan kepentingannya. Seiring waktu, daftar risiko akan berubah karena beberapa item dihapus dan yang lainnya muncul. Manajer proyek yang baik bekerja keras untuk menjaga risiko agar tidak berdampak pada jadwal dan biaya yang terkait dengan proyek.

	Penilaian Risiko
RISIKO 1:	Pengembangan sistem ini kemungkinan akan sangat lambat karena anggota tim proyek belum pernah memprogram di Java sebelum proyek ini.
Kemungkinan risiko:	Probabilitas risiko tinggi
Potensi dampak pada proyek:	Risiko ini mungkin akan meningkatkan waktu untuk menyelesaikan tugas pemrograman sebesar 50 persen.
Cara untuk mengatasi risiko ini:	

Hal ini sangat penting bahwa waktu dan sumber daya dialokasikan untuk pelatihan di awal mengenai Java untuk programmer yang digunakan dalam proyek ini. Pelatihan yang memadai akan mengurangi kurva pembelajaran awal Java saat pemrograman dimulai. Selain itu, keahlian di luar Java harus dimiliki setidaknya untuk beberapa bagian dari tugas pemrograman awal. Orang ini harus mampu memberikan pengetahuan pengalaman kepada tim proyek sehingga masalah terkait Java (yang tidak diketahui oleh programmer Java pemula) dapat diatasi.
RISIKO 2: …

GAMBAR 2-20 Contoh Penilaian Risiko

2.8 MENGATUR STAFF PROYEK

Penetapan staf proyek antara lain menentukan berapa banyak orang yang harus ditugaskan ke proyek, mencocokkan keterampilan orang dengan kebutuhan proyek, memotivasi mereka untuk memenuhi tujuan proyek, dan meminimalkan konflik yang akan terjadi seiring waktu. Hasil untuk bagian manajemen proyek ini adalah rencana kepegawaian, yang menggambarkan jumlah dan jenis orang yang akan bekerja pada proyek, struktur pelaporan keseluruhan, dan piagam proyek, yang menjelaskan tujuan dan aturan proyek. Namun, sebelum menjelaskan mengenai pengembangan rencana kepegawaian, cara memotivasi orang, dan cara menangani konflik, kami akan menjelaskan mengenai serangkaian karakteristik tim yang solid.

Karakteristik Tim yang Solid

Ide mengenai tim yang solid sudah ada sejak lama. Sebagian besar (jika tidak semua) kelompok siswa tidak mewakili gagasan tim yang solid, dan Anda mungkin tidak pernah memiliki kesempatan untuk menghargai keefektifan tim yang sebenarnya. Biasanya, dalam proyek kelas, siswa ditugaskan atau diminta untuk membentuk kelompok, yang mengakibatkan kemampuan untuk mengembangkan tim menjadi sangat terbatas. Namun, tim pengembangan yang berkembang sangat penting dalam pengembangan sistem informasi. Seluruh rangkaian pendekatan pengembangan perangkat lunak yang gesit bergantung pada tim yang berkembang pesat. Jika tidak, pendekatan pengembangan agile akan gagal total.

Menurut DeMarco dan Lister, “Tim yang solid adalah sekelompok orang yang menyatu begitu kuat sehingga tampak lebih besar daripada jumlah yang sebenarnya. Produksi tim semacam itu lebih besar daripada produksi orang yang sama yang bekerja dalam bentuk yang tidak solid.” Mereka melanjutkan dengan menyatakan bahwa "tim bisa menjadi hampir tak terbendung, menjadi raksasa untuk sukses." Kapan terakhir kali Anda bekerja dengan kelompok pada proyek kelas yang dapat digambarkan sebagai "raksasa untuk sukses"? Demarco dan Lister mengidentifikasi lima karakteristik tim yang solid.

Pertama, tim solid memiliki omset yang sangat rendah selama proyek. Biasanya, anggota tim solid merasa bertanggung jawab kepada anggota tim lainnya. Tanggung jawab ini dirasakan begitu kuat sehingga jika seorang anggota berniat meninggalkan tim, anggota tersebut akan merasa bahwa mereka mengecewakan tim dan bahwa mereka akan melanggar ikatan kepercayaan.

Kedua, tim solid memiliki rasa identitas yang kuat. Di banyak kelas, ketika Anda menjadi bagian dari sebuah grup, grup tersebut memilih beberapa nama yang lucu untuk mengidentifikasi grup dan membedakannya dari grup lain. Namun, dalam hal ini, bukan hanya pemilihan nama. Tetapi mengembangkan setiap anggota menjadi sesuatu yang terdiri dari hanya orang yang ada di dalam tim. Hal ini dapat dilihat ketika anggota tim cenderung

melakukan kegiatan yang tidak berhubungan dengan pekerjaan, misalnya makan siang bersama sebagai tim atau membentuk tim bola basket yang hanya terdiri dari anggota tim pengembangan.

Ketiga, rasa identitas yang kuat cenderung membuat tim merasa elit. Anggota tim pengembangan yang solid hampir memiliki kesombongan mengenai cara mereka berhubungan dengan karyawan non-tim. Contoh bagus yang muncul dalam pikiran yang memiliki rasa elit di luar lingkup tim pengembangan sistem informasi adalah tim olahraga tertentu, tim Navy Seal AS, atau tim SWAT kepolisian kota besar. Dalam ketiga contoh itu, setiap anggota tim sangat kompeten di bidang spesialisasinya, dan setiap anggota tim lainnya tahu (tidak berpikir) bahwa dia dapat bergantung pada anggota tim yang melakukan pekerjaan individunya dengan tingkat yang sangat keterampilan tinggi.

Keempat, selama proses pengembangan, tim solid merasa bahwa tim memiliki sistem informasi yang sedang dikembangkan dan bukan sebagai satu anggota individu. Dalam banyak hal, Anda hampir bisa mengatakan bahwa tim yang solid sedikit bersifat komunis. Dengan ini kami maksudkan bahwa kontribusi individu terhadap usaha tidak penting bagi tim yang sebenarnya. Satu-satunya hal yang penting adalah output dari tim. Namun, ini tidak berarti bahwa seorang anggota yang tidak berkontribusi dengan adil tidak luput dari hukuman. Dalam tim solid, setiap anggota yang tidak berproduksi sebenarnya melanggar ikatan kepercayaannya dengan anggota tim lainnya (lihat karakteristik pertama).

Ciri terakhir dari tim solid adalah anggota tim sangat menikmati (bersenang-senang) melakukan pekerjaan mereka. Para anggota sebenarnya suka pergi bekerja dan bergabung bersama anggota tim mereka. Hal ini dapat dikaitkan dengan tingkat tantangan yang mereka terima. Jika proyek ini menantang dan anggota tim akan belajar sesuatu dari proyek tsb, anggota tim yang solid akan senang menangani proyek tersebut.

Ketika sebuah tim bekerja, mereka akan menghindari lima disfungsi tim yang didefinisikan oleh Lencioni. Kurangnya kepercayaan adalah penyebab utama tim menjadi tidak berfungsi. Lencioni menjelaskan empat penyebab lain dari sebuah tim menjadi disfungsional yang bisa berasal dari kurangnya kepercayaan. Pertama, tim yang disfungsional takut akan konflik, sedangkan anggota tim solid tidak pernah takut akan konflik. Dalam tim solid, jika Anda tidak tahu bagaimana melakukan sesuatu bukanlah suatu masalah besar. Bahkan, tim ini menyediakan metode bagi anggota tim untuk membantu, yang akan meningkatkan tingkat kepercayaan di antara kedua anggota. Kedua, tim disfungsional tidak memiliki komitmen terhadap anggota individu. Sebaliknya, mereka cenderung fokus pada kinerja individu mereka daripada kinerja tim. Hal ini bahkan dapat merugikan tim pengembangan. Jelas, ini bukan masalah untuk tim solid. Ketiga, tim yang disfungsional mencoba menghindari akuntabilitas. Dengan tim yang kental, akuntabilitas tidak menjadi masalah. Anggota tim solid merasakan tingkat tanggung jawab yang tinggi kepada anggota tim lainnya. Tidak ada anggota tim yang ingin mengecewakan tim. Terlebih lagi, karena ikatan yang menyatukan tim solid, tidak ada anggota yang memiliki masalah dengan meminta pertanggungjawaban anggota lain atas kinerja mereka (atau kurangnya kinerja). Keempat, tim yang disfungsional tidak memperhatikan hasil tim. Sekali lagi, dalam hal ini, penyebab disfungsi ini adalah bahwa anggota individu hanya fokus pada tujuan individu mereka. Dari perspektif manajemen tim, pemimpin tim harus fokus untuk menyelaraskan tujuan tim; tim yang solid akan mencapai tujuan.

Rencana Kepegawaian

Langkah pertama untuk penempatan staf adalah menentukan jumlah rata-rata staf yang dibutuhkan untuk proyek tersebut. Untuk menghitung angka ini, bagilah total kerja orang-bulan dengan jadwal optimal. Jadi, untuk menyelesaikan proyek empat puluh orang-bulan

dalam sepuluh bulan, tim harus memiliki rata-rata empat anggota staf penuh waktu, meskipun ini dapat berubah dari waktu ke waktu jika spesialis yang berbeda masuk dan meninggalkan tim (misalnya, analis bisnis, programmer, penulis teknis).

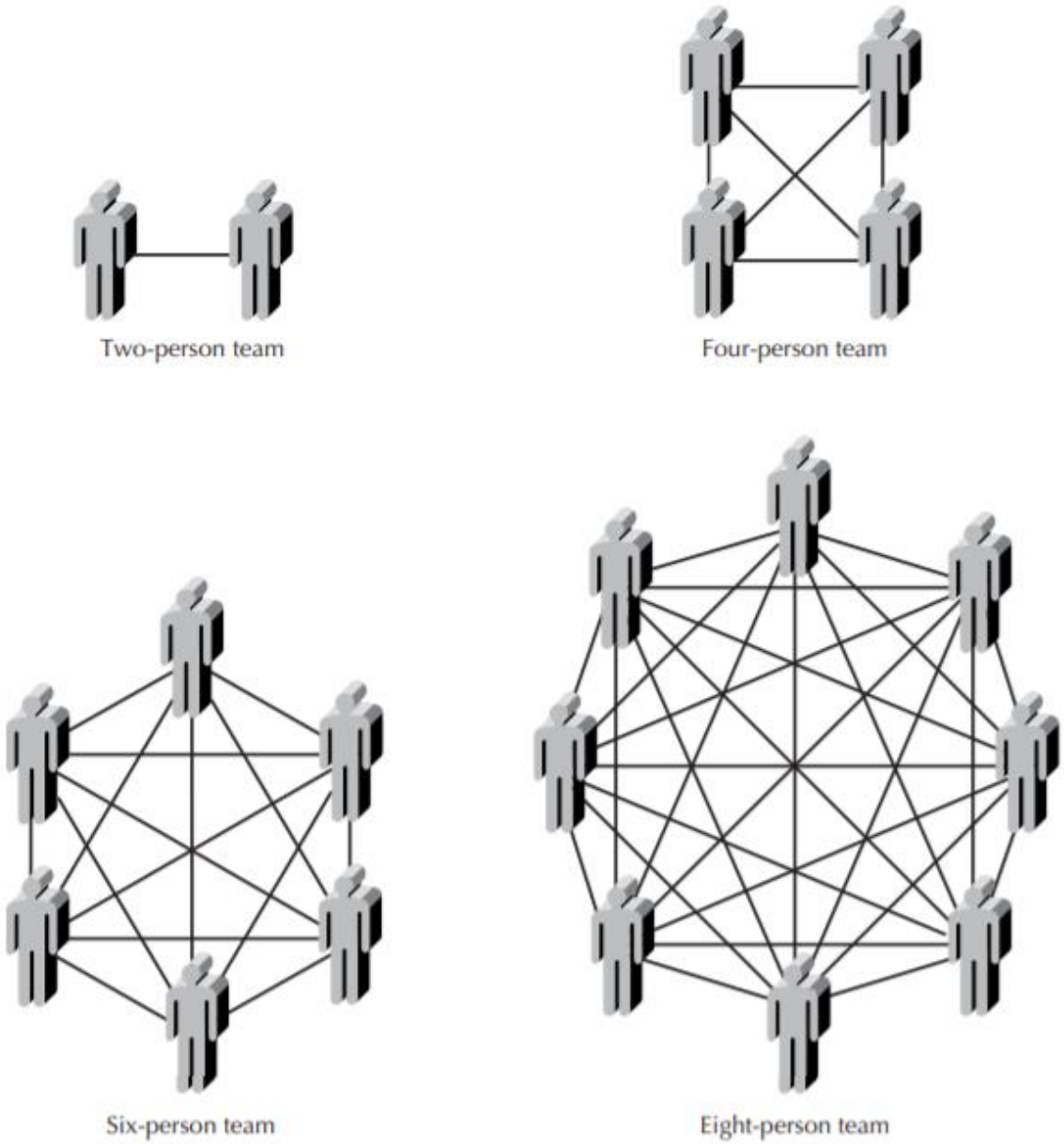
Sering kali, godaannya adalah menugaskan lebih banyak staf ke sebuah proyek untuk mempersingkat durasi proyek, tetapi ini bukan langkah yang bijaksana. Menambahkan sumber daya staf tidak berarti meningkatkan produktivitas; ukuran staf dan produktivitas tidak memiliki hubungan yang proporsional, terutama karena lebih sulit untuk mengoordinasikan sejumlah besar anggota staf. Semakin banyak tim tumbuh, semakin sulit untuk dikelola. Bayangkan betapa mudahnya bekerja dalam tim proyek yang terdiri dari dua orang: Anggota tim berbagi satu jalur komunikasi. Tetapi menambahkan dua orang meningkatkan jumlah jalur komunikasi menjadi enam, dan peningkatan yang lebih besar menghasilkan keuntungan yang lebih drastis dalam kompleksitas komunikasi. Gambar 2-21 mengilustrasikan dampak penambahan anggota tim ke tim proyek.

Salah satu cara untuk mengurangi penurunan efisiensi dalam tim adalah dengan memahami kompleksitas yang ada dalam jumlah dan membangun struktur pelaporan yang dapat mengurangi efeknya. Aturan umumnya adalah dengan menjaga ukuran tim kurang dari delapan sampai sepuluh orang; dan jika lebih banyak orang dibutuhkan, buatlah sub-tim. Dengan cara ini, manajer proyek dapat menjaga agar komunikasi tetap efektif dalam tim kecil, yang pada gilirannya dapat berkomunikasi dengan kontak di tingkat yang lebih tinggi dalam proyek.

Setelah manajer proyek memahami berapa banyak orang yang dibutuhkan untuk proyek tersebut, dia membuat rencana kepegawaian yang mencantumkan peran dan struktur pelaporan yang diusulkan yang diperlukan untuk proyek tersebut. Biasanya, sebuah proyek memiliki satu manajer proyek yang mengawasi kemajuan keseluruhan pengembangan, dengan inti tim yang terdiri dari berbagai jenis analis yang dijelaskan dalam bab sebelumnya. Seorang pemimpin fungsional biasanya ditugaskan untuk mengelola sekelompok analis, dan seorang pemimpin teknis mengawasi kemajuan sekelompok programmer dan lebih banyak anggota staf teknis.

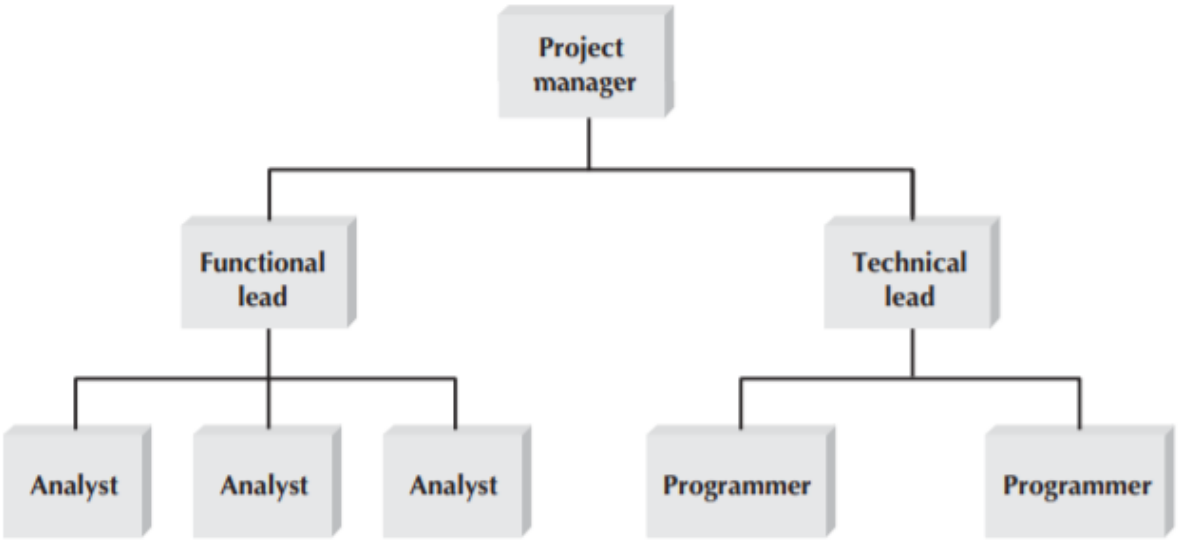
Ada banyak struktur untuk tim proyek; Gambar 2-22 mengilustrasikan satu kemungkinan konfigurasi tim proyek. Setelah peran ditentukan dan strukturnya ada, manajer proyek perlu memikirkan orang yang dapat mengisi setiap peran. Seringkali, satu orang mengisi lebih dari satu peran dalam tim proyek.

Saat Anda membuat tugas, ingatlah bahwa orang memiliki keterampilan teknis dan keterampilan interpersonal, dan keduanya penting dalam sebuah proyek. Keterampilan teknis berguna ketika mereka bekerja dengan tugas-tugas teknis (misalnya, pemrograman di Java) dan dalam mencoba memahami berbagai peran yang dimainkan teknologi dalam proyek tertentu (misalnya, bagaimana server web harus dikonfigurasi berdasarkan jumlah proyeksi hits dari pelanggan). Keterampilan interpersonal antara lain termasuk kemampuan interpersonal dan komunikasi yang digunakan ketika berhadapan dengan pengguna bisnis, eksekutif manajemen senior, dan anggota lain dari tim proyek. Kemampuan itu sangat penting ketika melakukan kegiatan pengumpulan persyaratan dan ketika menangani masalah kelayakan organisasi. Setiap proyek membutuhkan keterampilan teknis dan interpersonal yang unik.



GAMBAR 2-21 Meningkatnya Kompleksitas dengan Tim yang Lebih Besar

Idealnya, peran proyek diisi dengan orang-orang yang memiliki keterampilan yang tepat untuk pekerjaan itu. Namun, orang yang paling sesuai dengan peran mungkin saat itu tidak tersedia; mereka mungkin sedang mengerjakan proyek lain, atau mereka mungkin memang tidak ada di perusahaan. Oleh karena itu, menugaskan anggota tim proyek benar-benar merupakan kombinasi dari menemukan orang dengan keahlian yang sesuai dan menemukan orang yang tersedia. Ketika keterampilan anggota tim proyek yang tersedia tidak sesuai dengan apa yang sebenarnya dibutuhkan oleh proyek, manajer proyek memiliki beberapa pilihan untuk memperbaiki situasi. Pertama, menarik orang dari proyek lain, dan mengacak sumber daya. Ini adalah pendekatan yang paling mengganggu dari perspektif organisasi. Pendekatan lain adalah dengan menggunakan bantuan dari luar—seperti konsultan atau kontraktor—untuk melatih anggota tim dan mengajari mereka langkah yang benar. Mentoring juga bisa menjadi pilihan; seorang anggota tim proyek dapat dikirim untuk mengerjakan proyek serupa lainnya sehingga dia dapat kembali dengan keterampilan untuk melamar pekerjaan saat ini.



GAMBAR 2-22 Kemungkinan Struktur Pelaporan

Motivasi

Menugaskan orang ke tugas saja tidak cukup; manajer proyek perlu memotivasi orang-orang untuk memastikan keberhasilan proyek. Motivasi disebut sebagai pengaruh nomor satu terhadap kinerja orang, tetapi menentukan bagaimana cara memotivasi tim sangat sulit. Anda mungkin berpikir bahwa manajer proyek yang baik akan memotivasi staf mereka dengan memberi mereka uang dan bonus, tetapi sebagian besar manajer proyek setuju bahwa ini adalah hal terakhir. Semakin sering manajer menghadiahi anggota tim dengan uang, semakin mereka mengharapkannya—dan seringkali motivasi moneter tidak akan berhasil. Pink telah menyarankan seperangkat prinsip yang harus diikuti untuk memotivasi individu di perusahaan abad kedua puluh satu ini. Di bagian ini, kami mengadaptasi sarannya ke tim pengembangan sistem informasi.

Pink menyarankan untuk menggunakan aturan waktu 20 persen untuk memotivasi individu. Aturan ini menunjukkan bahwa 20 persen dari waktu karyawan harus dihabiskan untuk beberapa ide yang dia yakini. Proyek tidak harus berhubungan dengan proyek yang sedang dikerjakan. Di permukaan, hal ini terdengar seperti buang-buang waktu, tetapi ide ini tidak boleh dibuang. Google Gmail dan Google News dikembangkan menggunakan aturan waktu 20 persen. Jika 20 persen terdengar terlalu tinggi, Pink menyarankan agar Anda mempertimbangkan 10 persen untuk memulai.

Dia merekomendasikan bahwa perusahaan harus bersedia untuk mendanai penghargaan kecil "Now That". Penghargaan ini diberikan sebagai tanda penghargaan kecil untuk melakukan pekerjaan yang hebat. Namun, penghargaan ini tidak diberikan oleh seorang manajer kepada seorang karyawan tetapi dari seorang karyawan kepada rekan kerja karyawan tersebut. Penghargaannya berupa uang, tetapi sangat kecil, biasanya Rp.750.000. Dengan demikian, mereka benar-benar tidak berhubungan dengan perspektif moneter. Namun, hal tersebut sangat relevan karena diberikan oleh salah satu rekan karyawan untuk menunjukkan bahwa beberapa tindakan yang dilakukan karyawan tersebut diapresiasi.

Pink mendukung gagasan untuk menerapkan tes kata ganti Robert Reich (Sekretaris Tenaga Kerja Presiden Clinton). Jika seorang karyawan (atau anggota tim) mengacu pada perusahaan (tim) sebagai "mereka", maka ada kemungkinan nyata bahwa karyawan tersebut merasa terlepas atau mungkin diasingkan. Di sisi lain, ketika karyawan menyebut perusahaan

sebagai "kami", mereka jelas merasa seperti bagian dari organisasi. Dari sisi tim, ini bisa menjadi indikasi bahwa tim sudah mulai solid.

Pink menyarankan agar manajemen secara berkala mempertimbangkan untuk membiarkan karyawan satu hari di mana mereka dapat mengerjakan apa pun yang mereka inginkan. Dalam beberapa hal, hal ini terkait dengan aturan 20 persen. Namun, hal itu tidak selalu membutuhkan satu hari dalam seminggu (20 persen), tetapi hal itu membutuhkan beberapa kiriman. Kiriman dapat berupa program utilitas baru yang dapat digunakan oleh banyak proyek yang berbeda, dapat berupa prototipe baru dari produk perangkat lunak baru, atau dapat berupa perbaikan untuk proses bisnis yang digunakan secara internal. Tujuannya adalah untuk memberi anggota tim kemampuan untuk fokus pada masalah yang menarik dan menantang yang mungkin (atau mungkin tidak) memberikan hasil bagi perusahaan. Terlepas dari itu, hal ini menunjukkan kepercayaan dan rasa hormat yang dimiliki perusahaan terhadap karyawannya.

Dia merekomendasikan bahwa manajer menghapus masalah kompensasi dari persamaan motivasi. Dengan ini, dia bermaksud bahwa semua karyawan harus dibayar dalam jumlah yang cukup sehingga penghargaan kompensasi bukan suatu masalah. Karyawan teknis dalam tim proyek jauh lebih termotivasi oleh pengakuan, pencapaian, pekerjaan itu sendiri, tanggung jawab, kemajuan, dan kesempatan untuk mempelajari keterampilan baru. Penghargaan keuangan yang sederhana, seperti kenaikan gaji yang dianggap tidak adil, sebenarnya dapat menurunkan motivasi keseluruhan tim dan kinerja keseluruhan yang menjadi lebih rendah.

Dia menganjurkan bahwa bos abad kedua puluh lima (pemimpin tim) harus bersedia menyerahkan kendali. Banyak dari pendekatan pengembangan agile memberikan saran serupa. Appelo menyarankan bahwa kebijakan pintu terbuka yang didukung oleh seorang pemimpin tim sebenarnya dapat merugikan diri sendiri. Dalam kasus tim pengembangan perangkat lunak, kebijakan pintu terbuka menyiratkan bahwa pemimpin tim memiliki pintu ruangan yang dibiarkan terbuka, sementara anggota tim individu yang miskin tidak memiliki kantor dengan pintu. Dalam hal ini, Appelo menyarankan agar pemimpin tim pindah dari kantor dengan pintu ke ruang bersama yang sama di mana tim tinggal. Salah satu ide Pink lainnya adalah agar pemimpin tim tidak menggunakan bahasa pengontrol seperti memberi tahu anggota tim bahwa dia “harus” melakukan sesuatu. Sebaliknya, pemimpin tim harus meminta anggota tim untuk "mempertimbangkan" atau "memikirkan" ide tersebut. Dalam beberapa hal, seorang pemimpin tim sejati tidak boleh menerima pujian untuk ide apa pun yang terkait dengan tim. Sebaliknya, seorang pemimpin tim harus memberikan saran dan mendorong anggota tim untuk mempertimbangkan ide-ide dan, yang paling penting, membiarkan anggota tim dan tim yang menerima pujian.

Pink memberikan bukti bahwa motivasi intrinsik sangat penting bagi pengetahuan pekerja abad kedua puluh satu. Pink menyarankan bahwa individu yang memotivasi secara intrinsik membutuhkan mereka di tingkat otonomi, mendukung mereka sedemikian rupa sehingga mereka dapat menguasai bidang keahlian mereka, dan mendorong mereka untuk mengejar proyek dengan suatu tujuan. Memberikan otonomi kepada anggota tim berkaitan dengan konsep kepercayaan tim yang solid. Pemimpin tim perlu memercayai anggota tim untuk memegang perangkat lunak yang menjadi tanggung jawab mereka. Mendukung anggota tim agar mereka dapat menguasai bidang keahliannya dapat sesederhana memberikan dukungan untuk menghadiri konferensi, seminar, dan sesi pelatihan yang berhubungan dengan bidang keahlian anggota tersebut. Hal ini juga dapat menyiratkan seperti menyediakan anggota tim dengan lingkungan pengembangan kelas atas. Misalnya, ketika membangun aplikasi visualisasi informasi dan virtual reality, lingkungan perangkat keras dan perangkat lunak khusus dapat mempermudah penguasaan teknologi untuk mengembangkan aplikasi.

Hari-hari ini, hal ini sangat penting bagi anggota tim untuk merasa bahwa apa yang mereka lakukan dapat memberikan perbedaan. Seorang pemimpin tim harus mendorong anggota tim untuk mengatasi masalah yang dapat berdampak pada kehidupan orang. Hal ini dapat dengan mudah dicapai melalui penggunaan aturan 20 persen.

Menangani Konflik

Komponen ketiga dari kepegawaian adalah mengorganisir proyek untuk meminimalkan konflik di antara anggota kelompok. Kekompakan kelompok (daya tarik yang dirasakan anggota terhadap kelompok dan anggota lain) berkontribusi lebih besar terhadap produktivitas daripada kemampuan atau pengalaman individu anggota proyek. Mendefinisikan dengan jelas peran dalam proyek dan meminta anggota tim bertanggung jawab atas tugas-tugas mereka adalah cara yang baik untuk mulai mengurangi potensi konflik pada suatu proyek. Beberapa manajer proyek mengembangkan piagam proyek, yang mencantumkan norma dan aturan dasar proyek. Misalnya, piagam dapat menjelaskan kapan tim proyek harus bekerja, kapan rapat staf akan diadakan, bagaimana kelompok akan berkomunikasi satu sama lain, dan apa prosedur untuk memperbarui rencana kerja saat tugas diselesaikan. Gambar 2-23 mencantumkan teknik tambahan yang dapat digunakan pada awal proyek untuk meminimalkan konflik.

2.9 MANAJEMEN LINGKUNGAN DAN INFRASTRUKTUR

Alur kerja manajemen lingkungan dan infrastruktur mendukung tim pengembangan selama proses pengembangan. Alur kerja lingkungan utamanya berkaitan dengan memilih set alat yang benar yang akan digunakan selama proses pengembangan dan mengidentifikasi set standar yang sesuai untuk diikuti selama proses pengembangan. Alur kerja manajemen infrastruktur berkaitan dengan pemilihan tingkat dan jenis dokumentasi yang sesuai yang akan dibuat selama proses pengembangan. Aktivitas lain yang terkait dengan alur kerja manajemen infrastruktur antara lain mengembangkan, memodifikasi, dan menggunakan kembali komponen, kerangka kerja, pustaka, dan pola yang telah ditentukan sebelumnya. Topik penggunaan kembali dibahas dalam bab-bab selanjutnya.

• Mendefinisikan rencana proyek dengan jelas.
• Memastikan bahwa tim memahami betapa pentingnya proyek bagi organisasi.
• Mengembangkan prosedur operasi yang terperinci dan mengomunikasikan ini kepada anggota tim.
• Mengembangkan piagam proyek.
• Mengembangkan komitmen jadwal sebelumnya.
• Memperkirakan prioritas lain dan kemungkinan dampaknya terhadap proyek.
Sumber: H. J. Thamhain dan D. L. Wilemon, “Conflict Management in Project Life Cycles,” Sloan Management Review (Spring 1975).

GAMBAR 2-23 Strategi Menghindari Konflik

Peralatan CASE

Computer-aided software engineering (CASE) adalah kategori perangkat lunak yang mengotomatiskan semua atau sebagian dari proses pengembangan. Beberapa paket perangkat lunak CASE digunakan utamanya untuk mendukung alur kerja analisis guna membuat diagram terintegrasi dari sistem dan untuk menyimpan informasi mengenai komponen sistem,

sedangkan alat yang lain mendukung alur kerja desain yang dapat digunakan untuk menghasilkan kode untuk tabel database dan fungsionalitas sistem. CASE tool lainnya berisi fungsionalitas yang mendukung tugas selama proses pengembangan sistem. CASE hadir dalam berbagai jenis dalam hal kompleksitas dan fungsionalitas, dan banyak alat bagus tersedia di pasar untuk mendukung pengembangan sistem berorientasi objek (misalnya, ArgoUml, Enterprise Architect, Poseidon, Visual Paradigm, dan IBM's Rational Rose).

Manfaat menggunakan CASE sangat banyak. Dengan CASE tool, tugas dapat diselesaikan dan diubah lebih cepat, dokumentasi pengembangan menjadi terpusat, dan informasi diilustrasikan melalui diagram, yang biasanya lebih mudah dipahami. Secara potensial, CASE dapat mengurangi biaya pemeliharaan, meningkatkan kualitas perangkat lunak, dan membiasakan disiplin. Beberapa tim proyek bahkan menggunakan CASE untuk menilai besarnya perubahan pada proyek. Banyak CASE tool modern yang mendukung pengembangan sistem berorientasi objek yang mendukung teknik pengembangan yang dikenal sebagai rekayasa bolak-balik. Rekayasa bolak-balik berguna tidak hanya pada pembuatan kode tetapi juga untuk rekayasa balik diagram UML dari kode. Dengan cara ini, sistem dapat berkembang melalui diagram dan melalui kode secara bolak-balik.

Tentu saja, seperti hal lainnya, CASE tidak boleh dianggap sebagai peluru perak untuk pengembangan proyek. CASE tool yang canggih adalah aplikasi kompleks yang membutuhkan pelatihan dan pengalaman yang signifikan untuk mencapai manfaat nyata. Pengalaman kami menunjukkan bahwa CASE adalah cara yang berguna untuk mendukung komunikasi dan berbagi diagram proyek dan spesifikasi teknis selama digunakan oleh developer terlatih yang telah menerapkan CASE pada proyek sebelumnya. Semua CASE tool menggunakan repositori CASE untuk menyimpan diagram, model, dan desain I/O dan untuk memastikan konsistensi di seluruh pengulangan.

Standar

Anggota tim proyek perlu bekerja sama, dan sebagian besar perangkat lunak manajemen proyek dan CASE tool mendukung mereka dengan memberikan akses kepada semua orang yang bekerja pada sistem. Namun, tanpa prosedur yang ditetapkan, kolaborasi dapat membingungkan. Lebih buruk lagi, orang kadang-kadang ditugaskan kembali di tengah-tengah proyek. Penting agar pengetahuan proyek mereka tidak hilang dengan pergantian dan bahwa pergantian mereka dapat dilakukan dengan cepat.

Salah satu cara untuk memastikan bahwa setiap orang melakukan tugas dengan cara yang sama dan mengikuti prosedur yang sama adalah dengan membuat standar yang harus diikuti oleh tim proyek. Standar dapat mencakup aturan formal untuk penamaan file, formulir yang harus dilengkapi ketika tujuan tercapai, dan pedoman pemrograman. Gambar 2-24 menunjukkan beberapa contoh jenis standar yang dapat dibuat oleh sebuah proyek. Ketika sebuah tim membentuk standar dan kemudian mengikutinya, proyek dapat diselesaikan lebih cepat karena koordinasi tugas menjadi tidak terlalu rumit.

Standar berguna paling baik ketika dibuat pada awal setiap fase utama proyek dan dikomunikasikan dengan jelas kepada seluruh tim proyek. Saat tim bergerak maju, standar baru ditambahkan jika perlu. Beberapa standar (misalnya, konvensi penamaan file, pelaporan status) diterapkan selama seluruh proses pengembangan, sedangkan yang lain (misalnya, pedoman pemrograman) hanya untuk tugas-tugas tertentu.

Dokumentasi

Pada akhirnya, selama fase awal alur kerja infrastruktur, tim proyek menetapkan standar dokumentasi yang baik yang mencakup informasi rinci tentang tugas Proses Terpadu.

Biasanya, standar untuk dokumentasi yang diperlukan ditetapkan oleh organisasi pengembangan. Tim pengembangan hanya perlu memastikan standar dokumentasi mana yang sesuai untuk proyek pengembangan sistem saat ini. Seringkali, dokumentasi disimpan dalam pengikat proyek yang berisi semua hasil dan semua komunikasi internal yang terjadi—histori proyek. Kabar baiknya adalah bahwa Proses Terpadu memiliki seperangkat dokumentasi standar yang diinginkan. Dokumentasi biasanya mencakup permintaan sistem, analisis kelayakan, versi asli dan yang lebih baru dari estimasi upaya, rencana kerja yang berkembang, dan diagram UML untuk model fungsional, struktural, dan perilaku.

Tipe Standar	Contoh
Standar dokumentasi	Tanggal dan nama proyek akan muncul sebagai tajuk di semua dokumentasi. Semua margin harus diatur 1 inci. Semua kiriman harus ditambahkan ke pengikat proyek dan dicatat dalam daftar isinya.
Standar coding	Semua modul kode harus menyertakan header yang mencantumkan programmer, tanggal terakhir pembaruan, dan deskripsi singkat tentang tujuan kode. Indentasi harus digunakan untuk menunjukkan perulangan, pernyataan if-then-else, dan pernyataan kasus. Rata-rata, setiap program harus menyertakan satu baris komentar untuk setiap lima baris kode.
Standar prosedural	Catat kemajuan tugas yang sebenarnya dalam rencana kerja setiap Senin jam 10 pagi. Laporkan ke rapat pembaruan proyek pada hari Jumat pukul 15:30. Semua perubahan pada dokumen persyaratan harus disetujui oleh manajer proyek.
Standar persyaratan spesifikasi	Nama program yang akan dibuat Deskripsi tujuan program Perhitungan khusus yang perlu dihitung Aturan bisnis yang harus dimasukkan ke dalam program Kode semu

	Batas tanggal terakhir
Standar desain antarmuka pengguna	Label akan muncul dalam teks tebal, rata kiri, dan diikuti oleh titik dua. Urutan tab layar akan berpindah dari kiri atas ke kanan bawah. Utama akselerator akan disediakan untuk semua bidang yang dapat diperbarui.

GAMBAR 2-24 Contoh Standar Proyek

Praktik manajemen proyek yang buruk biasanya baru akan melakukan hal ini hingga menit terakhir; ini biasanya mengarah ke sistem tidak berdokumen yang tidak dipahami oleh siapa pun. Tim proyek yang baik belajar untuk mendokumentasikan histori sistem saat berkembang sementara detailnya masih segar dalam ingatan mereka. Di sebagian besar CASE tool yang mendukung pengembangan sistem berorientasi objek, beberapa dokumentasi dapat diotomatisasi. Misalnya, jika bahasa pemrograman yang dipilih untuk mengimplementasikan sistem adalah Java, maka dimungkinkan untuk membuat halaman manual HTML secara otomatis yang akan menjelaskan kelas yang diimplementasikan. Hal ini dilakukan melalui alat javadoc yang merupakan bagian dari lingkungan pengembangan Java. Alat lain yang memungkinkan pengembang secara otomatis yang menghasilkan dokumentasi HTML untuk diagram UML, misalnya, umldoc, yang merupakan bagian dari Poseidon untuk alat UML CASE. Meskipun hampir semua developer benci membuat dokumentasi dan dokumentasi membutuhkan waktu yang berharga, ini adalah investasi yang baik yang akan terbayar dalam jangka panjang.

Tips Praktis Menghindari Kesalahan Perencanaan Klasik

Seperti yang ditunjukkan oleh David Umphress dari Universitas Seattle, menyaksikan sebagian besar organisasi mengembangkan sistem seperti menonton tayangan ulang Gilligan's Island. Pada awal setiap episode, seseorang datang dengan skema cockamamie untuk keluar dari pulau, dan tampaknya berhasil untuk sementara waktu, tetapi ada yang tidak beres dan para mereka mendapati diri mereka kembali ke tempat awal—terjebak di pulau itu. Demikian pula, sebagian besar perusahaan memulai proyek baru dengan ide-ide besar yang tampaknya berhasil, hanya untuk membuat kesalahan klasik dan menyelesaikan proyek melebihi jadwal jadwal, melebihi anggaran, atau keduanya. Di sini kami merangkum empat kesalahan klasik dalam aspek perencanaan dan manajemen proyek dan mendiskusikan bagaimana menghindarinya:

1. ***Jadwal yang terlalu optimis***: Pemikiran yang penuh harapan dapat menyebabkan jadwal yang terlalu optimis sehingga membuat analisis dan desain dipersingkat (persyaratan utama tidak ada) dan memberikan tekanan kuat pada programmer, yang menghasilkan coding yang buruk (penuh dengan bug).
   Solusi: Jangan menaikkan perkiraan waktu; sebagai gantinya, secara eksplisit menjadwalkan waktu longgar pada akhir setiap fase untuk memperhitungkan variabilitas dalam perkiraan.
2. ***Gagal memantau jadwal:*** Jika tim tidak melaporkan kemajuan secara berkala, sehingga membuat semuanya tidak ada yang tahu apakah proyek sesuai jadwal.
   Solusi: Minta anggota tim untuk melaporkan kemajuan (atau kurangnya perkembangan sekecil apapun) secara jujur setiap minggu. Tidak ada hukuman untuk melaporkan kurangnya kemajuan, tetapi ada sanksi langsung untuk laporan yang menyesatkan.

3. ***Gagal memperbarui jadwal***: Ketika bagian dari jadwal tertinggal (misalnya, pengumpulan informasi menggunakan semua slack dalam item 1 ditambah 2 minggu), tim proyek sering berpikir bahwa mereka dapat mengganti waktu dengan bekerja lebih cepat nanti. Tidak bisa. Ini adalah peringatan dini bahwa seluruh jadwal terlalu optimis.
   Solusi: Segera revisi jadwal dan beri tahu sponsor proyek tentang tanggal akhir yang baru atau gunakan timeboxing untuk mengurangi fungsionalitas atau memindahkannya ke versi mendatang.
4. ***Menambahkan orang ke proyek yang terlambat***: Ketika sebuah proyek lewat dari jadwalnya, godaannya adalah menambahkan lebih banyak orang untuk mempercepatnya. Hal ini membuat proyek memakan waktu lebih lama karena meningkatnya masalah koordinasi dan membutuhkan staf untuk meluangkan waktu untuk bisa menjelaskan apa yang telah dilakukan.
   Solusi: Revisi jadwal, gunakan timeboxing, buang coding yang berisi bug, dan tambahkan orang hanya untuk mengerjakan bagian proyek yang terisolasi.

Pertanyaan

1. Berikan tiga contoh kebutuhan bisnis untuk sebuah sistem.
2. Apa tujuan dari komite persetujuan? Siapa yang biasanya berada pada komite ini?
3. Mengapa permintaan sistem harus dibuat oleh pelaku bisnis sebagai lawan dari profesional IS?
4. Apa perbedaan antara nilai tidak berwujud dan nilai berwujud? Berikan masing-masing tiga contoh.
5. Apa tujuan dari permintaan sistem dan analisis kelayakan? Bagaimana mereka digunakan dalam proses pemilihan proyek?
6. Jelaskan dua masalah khusus yang mungkin penting untuk dicantumkan pada permintaan sistem.
7. Jelaskan tiga teknik untuk analisis kelayakan.
8. Jelaskan proyek berisiko dalam hal kelayakan teknis. Jelaskan proyek yang tidak akan dianggap berisiko.
9. Apa langkah-langkah untuk menilai kelayakan ekonomi? Jelaskan setiap langkah.
10. Sebutkan dua manfaat tidak berwujud. Jelaskan bagaimana manfaat ini dapat diukur.
11. Sebutkan dua mafaat nyata dan dua biaya operasional untuk suatu sistem. Bagaimana Anda menentukan nilai yang harus ditetapkan untuk setiap item?
12. Jelaskan nilai sekarang bersih dan laba atas investasi untuk analisis biaya-manfaat. Mengapa perhitungan ini digunakan?
13. Apa titik impas untuk proyek tersebut? Bagaimana cara menghitungnya?
14. Apa itu analisis pemegang kepentingan? Diskusikan tiga pemegang kepentingan yang relevan untuk sebagian besar proyek.
15. Mengapa banyak proyek berakhir dengan tenggat waktu yang tidak masuk akal? Bagaimana seharusnya seorang manajer proyek bereaksi terhadap tuntutan yang tidak masuk akal?
16. Apa trade-off yang harus dikelola oleh manajer proyek?
17. Bandingkan dan kontraskan Gantt chart dengan diagram jaringan.
18. Beberapa perusahaan menyewa perusahaan konsultan untuk mengembangkan rencana proyek awal dan mengelola proyek tetapi menggunakan analis dan pemrogram mereka sendiri untuk mengembangkan sistem. Menurut Anda mengapa beberapa perusahaan melakukan ini?
19. Apa yang dimaksud dengan titik Use-Case? Untuk apa itu digunakan?
20. Proses apa yang kita gunakan untuk memperkirakan pengembangan sistem berdasarkan Use-Case?

21. Sebutkan dua cara untuk mengidentifikasi tugas-tugas yang perlu diselesaikan selama proyek berlangsung.
22. Masalah apa saja yang terkait dengan WBS konvensional?
23. Apa itu WBS evolusioner? Bagaimana cara mengatasi masalah yang terkait dengan WBS konvensional?
24. Apa itu rencana kerja berulang?
25. Apa itu scope creep, dan bagaimana cara mengelolanya?
26. Apa itu timeboxing, dan mengapa digunakan?
27. Buat daftar risiko potensial yang dapat memengaruhi hasil proyek.
28. Jelaskan perbedaan antara petunjuk teknis dan petunjuk fungsional. Bagaimana mereka mirip?
29. Jelaskan tiga keterampilan teknis dan tiga keterampilan interpersonal yang sangat penting untuk dimiliki dalam setiap proyek.
30. Apa cara terbaik untuk memotivasi tim? Apa cara terburuk?
31. Sebutkan tiga teknik untuk mengurangi konflik.
32. Jelaskan tiga jenis standar dan berikan contoh masing-masing.
33. Apa yang termasuk dalam pengikat proyek? Bagaimana pengikat proyek diatur?

Latihan

A. Cari artikel berita di majalah perdagangan TI (misalnya, Computerworld) tentang organisasi yang menerapkan sistem komputer baru. Jelaskan nilai berwujud dan tidak berwujud yang mungkin diwujudkan organisasi dari sistem baru.
B. Dealer mobil telah menyadari betapa menguntungkannya menjual mobil menggunakan Web. Berpura-puralah bahwa Anda bekerja untuk dealer mobil lokal yang merupakan bagian dari rantai besar seperti Toyota. Buat permintaan sistem yang mungkin Anda gunakan untuk mengembangkan sistem penjualan berbasis Web. Ingatlah untuk membuat daftar masalah khusus yang relevan dengan proyek.
C. Misalkan Anda tertarik untuk membeli komputer baru. Buat analisis biaya-manfaat yang menggambarkan laba atas investasi yang akan Anda terima dari melakukan pembelian ini. Situs web terkait komputer (mis., Apple, Dell, HP, lenovo atau lainnya) harus memiliki biaya nyata yang dapat Anda sertakan dalam analisis Anda. Proyeksikan angka Anda untuk memasukkan periode tiga tahun dan berikan nilai sekarang bersih dari total akhir.
D. Situs web Amazon.com awalnya menjual buku; kemudian manajemen perusahaan memutuskan untuk memperluas sistem berbasis Web mereka untuk memasukkan produk lain. Bagaimana Anda menilai kelayakan usaha ini ketika ide pertama kali muncul? Seberapa berisikokah Anda mempertimbangkan proyek yang menerapkan ide ini? Mengapa?
E. Wawancarai seseorang yang bekerja di sebuah organisasi besar dan minta dia untuk menjelaskan proses persetujuan yang ada untuk menyetujui proyek pengembangan baru. Apa yang mereka pikirkan tentang prosesnya? Apa masalahnya? Apa manfaatnya?
F. Kunjungi situs web manajemen proyek, seperti Project Management Institute (www.pmi.org). Sebagian besar memiliki tautan ke produk perangkat lunak manajemen proyek, kertas putih, dan penelitian. Periksa beberapa tautan untuk manajemen proyek untuk lebih memahami berbagai situs Internet yang berisi informasi yang terkait dengan bab ini.
G. Pilih topik manajemen proyek tertentu seperti KASUS, perangkat lunak manajemen proyek, atau timeboxing dan cari informasi tentang topik itu menggunakan Web. Mesin pencari apa pun (mis., Bing, Google) dapat memberikan titik awal untuk upaya Anda.

H. Berpura-puralah bahwa kantor layanan karir di universitas Anda ingin mengembangkan sistem yang mengumpulkan resume mahasiswa dan membuatnya tersedia bagi mahasiswa dan perekrut melalui Web. Mahasiswa harus dapat memasukkan informasi resume mereka ke dalam template resume standar. Informasi tersebut kemudian disajikan dalam format resume, dan juga ditempatkan dalam database yang dapat ditanyakan menggunakan formulir pencarian online. Anda telah ditugaskan untuk proyek tersebut. Kembangkan rencana untuk memperkirakan proyek. Menurut Anda, berapa lama waktu yang dibutuhkan bagi Anda dan tiga siswa lainnya untuk menyelesaikan proyek tersebut? Berikan dukungan untuk jadwal yang Anda usulkan.

I. Lihat situasi dalam latihan H. Anda telah diberitahu bahwa musim perekrutan dimulai sebulan dari hari ini dan bahwa sistem baru harus digunakan. Bagaimana Anda akan mendekati situasi ini? Jelaskan apa yang dapat Anda lakukan sebagai manajer proyek untuk memastikan bahwa tim Anda tidak kelelahan karena tenggat waktu dan komitmen yang tidak masuk akal.

J. Pertimbangkan sistem yang dijelaskan dalam latihan H. Buat rencana kerja yang mencantumkan tugas yang perlu diselesaikan untuk memenuhi tujuan proyek. Buat bagan Gantt dan diagram jaringan dalam alat manajemen proyek (mis., Microsoft Project) atau gunakan paket spreadsheet untuk menampilkan tugas tingkat tinggi proyek secara grafis.

K. Misalkan Anda bertanggung jawab atas proyek yang dijelaskan dalam latihan H dan proyek tersebut akan dikelola oleh anggota kelas Anda. Apakah teman sekelas Anda memiliki semua keterampilan yang tepat untuk mengimplementasikan proyek semacam itu? Jika tidak, bagaimana Anda akan memastikan bahwa keterampilan yang tepat tersedia untuk menyelesaikan pekerjaan?

L. Lengkapi lembar kerja poin Use-Case untuk memperkirakan upaya membangun sistem yang dijelaskan dalam latihan H, I, J, dan K. Anda perlu membuat asumsi mengenai aktor, Use-Case, dan kompleksitas teknis serta faktor lingkungan.

M. Pertimbangkan aplikasi yang digunakan di sekolah Anda untuk mendaftar kelas. Lengkapi lembar kerja titik Use-Case untuk memperkirakan upaya untuk membangun aplikasi semacam itu. Anda perlu membuat beberapa asumsi tentang antarmuka aplikasi dan berbagai faktor yang mempengaruhi kompleksitasnya.

N. Berpura-pura bahwa instruktur Anda telah meminta Anda dan dua teman untuk membuat halaman Web untuk menjelaskan kursus kepada calon siswa dan memberikan informasi kelas terkini (misalnya, silabus, tugas, bacaan) kepada siswa saat ini. Anda telah diberi peran sebagai pemimpin, jadi Anda perlu mengoordinasikan aktivitas Anda dan aktivitas teman sekelas Anda hingga proyek selesai. Jelaskan bagaimana Anda akan menerapkan teknik manajemen proyek yang telah Anda pelajari dalam bab ini dalam situasi ini. Sertakan deskripsi tentang bagaimana Anda akan membuat rencana kerja, staf proyek, dan mengoordinasikan semua aktivitas—Anda dan teman sekelas Anda

O. Pilih dua paket perangkat lunak manajemen proyek dan teliti mereka menggunakan Web atau majalah perdagangan. Jelaskan fitur dari kedua paket tersebut. Jika Anda seorang manajer proyek, mana yang akan Anda gunakan untuk membantu mendukung pekerjaan Anda? Mengapa?

P. Pada tahun 1997, Oxford Health Plans mengalami masalah komputer yang menyebabkan perusahaan melebih-lebihkan pendapatan dan meremehkan biaya medis. Masalah disebabkan oleh migrasi sistem pemrosesan klaimnya dari sistem operasi Pick ke sistem berbasis UNIX yang menggunakan perangkat lunak dan perangkat keras database Oracle dari Pyramid Technology. Akibatnya, harga saham Oxford anjlok, dan memperbaiki sistem menjadi prioritas nomor satu bagi perusahaan. Misalkan Anda ditugaskan untuk mengelola perbaikan sistem pemrosesan klaim. Jelas,

tim proyek tidak akan bersemangat. Bagaimana Anda akan memotivasi anggota tim untuk memenuhi tujuan proyek?

BAGIAN SATU

PEMODELAN ANALISIS

Pemodelan analisis menjawab pertanyaan tentang siapa yang akan menggunakan sistem, apa yang akan dilakukan sistem, dan di mana dan kapan akan digunakan. Selama analisis, persyaratan rinci diidentifikasi dan proposal sistem dibuat. Tim kemudian memiliki model fungsional (diagram Use-Case, diagram aktivitas, dan deskripsi Use-Case), model struktural (kartu CRC dan diagram kelas, dan diagram objek), dan model perilaku (diagram urutan, diagram komunikasi, status perilaku mesin, dan matriks CRUDE).

BAB 3

PENENTUAN PERSYARATAN

Salah satu kegiatan pertama seorang analis adalah menentukan kebutuhan bisnis untuk sistem baru. Bab ini dimulai dengan menyajikan definisi persyaratan, yaitu suatu dokumen yang mencantumkan kemampuan sistem baru. Kemudian menjelaskan bagaimana menganalisis persyaratan menggunakan strategi analisis persyaratan dan bagaimana mengumpulkan persyaratan menggunakan wawancara, sesi JAD, kuesioner, analisis dokumen, dan observasi. Bab ini juga menjelaskan mengenai seperangkat teknik dokumentasi persyaratan alternatif dan menjelaskan mengenai dokumen proposal sistem yang menyatukan semuanya.

3.1 TUJUAN

- Memahami cara membuat definisi persyaratan
- Familiar dengan teknik analisis kebutuhan
- Memahami kapan harus menggunakan setiap teknik analisis kebutuhan
- Memahami cara mengumpulkan persyaratan menggunakan wawancara, sesi JAD, kuesioner, analisis dokumen, dan observasi
- Memahami penggunaan peta konsep, kartu cerita, dan daftar tugas sebagai teknik dokumentasi persyaratan
- Memahami kapan harus menggunakan setiap teknik pengumpulan persyaratan
- Mampu membuat proposal sistem

3.2 PENDAHULUAN

Proses pengembangan sistem dapat membantu organisasi dalam berpindah dari sistem saat ini (sering disebut sistem sekarang) ke sistem baru (sering disebut sistem yang akan datang). Keluaran perencanaan, yang dibahas dalam sebelumnya, adalah permintaan sistem, yang memberikan gagasan umum untuk sistem yang akan dibuat, mendefinisikan ruang lingkup proyek, dan menyediakan rencana kerja awal. Analisis menghasilkan ide-ide umum dalam permintaan sistem dan menyempurnakannya menjadi definisi kebutuhan rinci (dijelaskan dalam bab ini), model fungsional, model struktural , dan model perilaku yang bersama-sama membentuk sistem usul yang akan dijelaskan di bab berikutnya. Usulan sistem juga mencakup hasil manajemen proyek yang direvisi, seperti analisis kelayakan dan rencana kerja (dijelaskan di bab sebelumnya).

Keluaran analisis, yaitu proposal sistem, dipresentasikan kepada komite persetujuan, yang akan memutuskan apakah proyek akan dilanjutkan. Jika disetujui, proposal sistem

diterapkan ke dalam desain, dan elemen-elemennya (definisi kebutuhan dan model fungsional, struktural, dan perilaku) digunakan sebagai masukan untuk langkah-langkah dalam desain. Tahap ini akan menyempurnakan dan mendefinisikan lebih detail mengenai bagaimana sistem akan dibangun.

Garis antara analisis dan desain sangat kabur. Hal ini karena hasil yang dibuat selama analisis benar-benar merupakan langkah pertama dalam desain sistem baru. Banyak keputusan desain utama untuk sistem baru ditemukan dalam hasil analisis. Penting untuk diingat bahwa hasil dari analisis benar-benar merupakan langkah pertama dalam desain sistem baru.

Dalam banyak hal, karena di sinilah elemen utama sistem pertama kali muncul, langkah penentuan persyaratan adalah langkah paling kritis dari keseluruhan proses pengembangan sistem. Selama penentuan persyaratan, sistem mudah diubah karena pekerjaan yang dilakukan baru sedikit. Saat sistem bergerak melalui proses pengembangan sistem, semakin sulit untuk kembali ke penentuan persyaratan dan membuat perubahan yang besar akibat adanya semua pengerjaan ulang. Beberapa studi telah menunjukkan bahwa lebih dari setengah dari semua kegagalan sistem adalah karena masalah dengan persyaratan. Inilah sebabnya mengapa pendekatan berulang dalam metodologi berorientasi objek sangat efektif—kelompok kecil persyaratan dapat diidentifikasi dan diimplementasikan secara bertahap, yang memungkinkan sistem secara keseluruhan untuk berkembang dari waktu ke waktu.

3.3 PENENTUAN PERSYARATAN

Tujuan dari penentuan persyaratan adalah untuk mengubah penjelasan yang memiliki tingkat yang sangat tinggi dari suatu persyaratan bisnis yang dinyatakan dalam permintaan sistem ke dalam daftar kebutuhan yang lebih tepat yang dapat digunakan sebagai masukan untuk analisis lainnya (membuat model fungsional, struktural, dan perilaku). Perluasan persyaratan pada akhirnya akan mengarah pada desain sistem.

Mendefinisikan Persyaratan

Persyaratan hanyalah pernyataan tentang apa yang harus dilakukan sistem atau karakteristik apa yang harus dimiliki. Selama analisis, persyaratan ditulis melalui perspektif pelaku bisnis, dan persyaratan itu fokus pada "apa" dari suatu sistem. Karena fokus pada kebutuhan pengguna bisnis, biasanya disebut sebagai persyaratan bisnis (dan terkadang disebut persyaratan pengguna). Kemudian dalam desain, kebutuhan bisnis berkembang menjadi lebih teknis, dan persyaratan itu menggambarkan bagaimana sistem akan diimplementasikan. Persyaratan dalam desain ditulis dari sudut pandang developer, dan biasanya disebut sebagai persyaratan sistem.

Kami ingin menekankan bahwa tidak ada garis hitam-putih yang memisahkan persyaratan bisnis dan persyaratan sistem—dan beberapa perusahaan menggunakan istilah tersebut secara bergantian. Hal penting untuk diingat adalah bahwa persyaratan adalah pernyataan tentang apa yang harus dilakukan sistem, dan persyaratan akan berubah seiring waktu saat proyek bergerak dari awal ke elaborasi lalu ke konstruksi. Persyaratan berkembang dari pernyataan rinci tentang kemampuan bisnis yang harus dimiliki suatu sistem menjadi pernyataan terperinci tentang bagaimana seccara teknis kemampuan tersebut akan diimplementasikan dalam sistem baru.

Persyaratan dapat berupa fungsional atau nonfungsional di lingkungan. Persyaratan fungsional berhubungan langsung dengan proses yang harus dilakukan sistem atau informasi yang harus dimilikinya. Misalnya, persyaratan yang menyatakan bahwa suatu sistem harus memiliki kemampuan untuk mencari inventaris yang tersedia atau untuk melaporkan pengeluaran aktual dan yang dianggarkan adalah persyaratan fungsional. Persyaratan

fungsional mengalir langsung ke pembuatan model fungsional, struktural, dan perilaku yang mewakili fungsionalitas sistem yang berkembang (lihat bab-bab berikutnya).

Persyaratan nonfungsional mengacu pada sifat perilaku yang harus dimiliki sistem, seperti kinerja dan kegunaan. Kemampuan untuk mengakses sistem menggunakan browser web dianggap sebagai persyaratan nonfungsional. Persyaratan nonfungsional dapat memengaruhi analisis lainnya (model fungsional, struktural, dan perilaku) tetapi seringkali secara tidak langsung; persyaratan nonfungsional digunakan utamanya dalam desain ketika keputusan dibuat mengenai database, antarmuka pengguna, perangkat keras dan perangkat lunak, dan arsitektur fisik yang mendasari sistem.

Persyaratan nonfungsional menggambarkan berbagai karakteristik mengenai sistem: operasional, kinerja, keamanan, dan budaya dan politik. Persyaratan operasional mengatasi masalah yang terkait dengan persyaratan fisik dan teknis di mana sistem akan beroperasi. Persyaratan kinerja mengatasi masalah yang terkait dengan kecepatan, kapasitas, dan reliabilitas sistem. Persyaratan keamanan mengatasi masalah yang berkaitan dengan siapa yang memiliki akses ke sistem dan dalam keadaan tertentu seperti apa. Persyaratan budaya dan politik berurusan dengan isu-isu yang berkaitan dengan budaya, faktor politik, dan persyaratan hukum yang mempengaruhi sistem. Karakteristik ini tidak menggambarkan proses bisnis atau informasi, tetapi sangat penting dalam memahami seperti apa sistem akhir yang seharusnya. Persyaratan nonfungsional sangat mempengaruhi keputusan yang akan dibuat selama desain sistem. Kita akan kembali ke topik ini nanti dalam buku ini ketika kita membahas desain (lihat bab-bab berikutnya).

Salah satu bidang pengembangan sistem informasi yang fokus pada perbedaan kebutuhan fungsional dan nonfungsional adalah kualitas perangkat lunak. Ada banyak model yang diusulkan untuk mengukur kualitas perangkat lunak. Namun, hampir semuanya membedakan kebutuhan fungsional dan nonfungsional. Dari perspektif kualitas, kualitas fungsional berkaitan dengan tingkat perangkat lunak dalam memenuhi persyaratan fungsional, yaitu, seberapa banyak masalah aktual dapat diselesaikan oleh perangkat lunak yang disediakan. Sedangkan kebutuhan nonfungsional berkaitan dengan dimensi efisiensi, perawatan, portabilitas, reliabilitas, reusabilitas, kemampuan pengujian, dan kualitas penggunaan. Sebagaimana dinyatakan di atas, dimensi terkait nonfungsional dihubungkan dengan desain rinci aktual dan implementasi sistem.

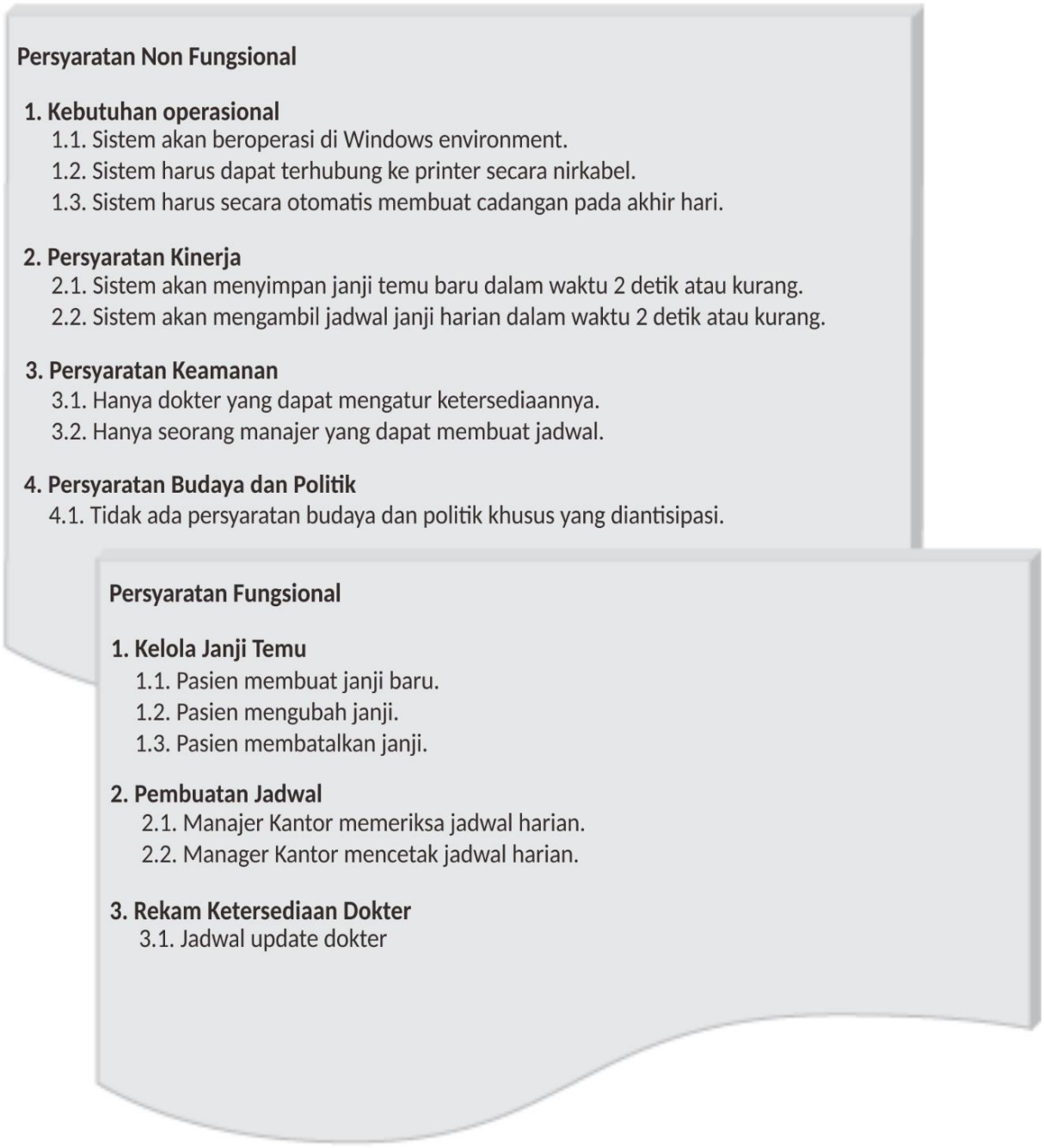
Saat mempertimbangkan penyesuaian ISO 9000, dimensi kualitas didekomposisi lebih lanjut menjadi dimensi yang dapat dilihat oleh pengguna (eksternal) dan dimensi yang tidak dapat dilihat oleh pengguna (internal). Dimensi nonfungsional eksternal meliputi efisiensi, reliabilitas, dan penggunaan, sedangkan dimensi nonfungsional internal meliputi pemeliharaan, portabilitas, penggunaan kembali, dan kemampuan pengujian. Dari perspektif pengguna, dimensi eksternal lebih penting. Jika sistem terlalu sulit untuk digunakan, terlepas dari seberapa baik sistem dalam memecahkan masalah, pengguna tidak akan menggunakan sistem. Dengan kata lain, dari sudut pandang pengguna, agar sistem informasi berhasil, sistem tidak hanya harus memenuhi spesifikasi fungsional, tetapi juga harus memenuhi spesifikasi nonfungsional eksternal. Dari perspektif developer, dimensi internal juga penting. Misalnya, mengingat bahwa sistem yang sukses cenderung berumur panjang dan multiplatform, baik dimensi perawatan dan portabilitas dapat memiliki implikasi strategis untuk sistem yang sedang dikembangkan. Selain itu, mengingat pendekatan pengembangan agile yang digunakan dalam industri saat ini, pengembangan perangkat lunak yang dapat digunakan kembali dan dapat diuji sangat penting.

Tiga topik tambahan yang mempengaruhi persyaratan sistem informasi adalah Undang-Undang Sarbanes-Oxley, penyesuaian COBIT (Control Objectives for Information and related

Technology/Tujuan Kontrol untuk Informasi dan Teknologi terkait) dan penyesuaian Model Kematangan Kemampuan. Tergantung pada sistem yang dipertimbangkan, ketiga topik ini dapat memengaruhi definisi persyaratan fungsional sistem, persyaratan nonfungsional, atau keduanya. Undang-Undang Sarbanes-Oxley, misalnya, mengamanatkan persyaratan fungsional dan nonfungsional tambahan. Hal ini termasuk masalah keamanan tambahan (nonfungsional) dan persyaratan informasi spesifik yang sekarang harus disediakan oleh manajemen (fungsional). Saat mengembangkan sistem informasi keuangan, developer sistem informasi harus memastikan untuk menyertakan keahlian Sarbanes-Oxley dalam tim pengembangan. Selain itu, klien dapat menuntut penyesuaian COBIT atau bahwa tingkat Model Kematangan Kemampuan tertentu telah dicapai agar perusahaan dapat dianggap sebagai vendor yang memungkinkan untuk memasok sistem yang sedang dipertimbangkan. Jelas, jenis persyaratan ini menambah persyaratan nonfungsional. Diskusi lebih lanjut tentang topik-topik ini tidak akan dibahas dalam buku ini.

Topik terbaru lainnya yang mempengaruhi persyaratan untuk beberapa sistem adalah globalisasi. Misalnya, rantai pasokan informasi global menghasilkan banyak persyaratan nonfungsional tambahan. Jika lingkungan operasional yang diperlukan tidak ada untuk solusi seluler yang akan dikembangkan, penting untuk menyesuaikan solusi dengan lingkungan lokal. Atau, mungkin tidak masuk akal untuk berharap penerapan solusi berbasis teknologi tinggi di area yang tidak memiliki infrastruktur daya dan komunikasi yang dibutuhkan. Dalam beberapa kasus, kita mungkin perlu mempertimbangkan untuk memberikan dukungan pada beberapa bagian dari rantai pasokan informasi global dengan sistem manual—bukan otomatis.

Sistem manual memiliki serangkaian persyaratan yang berbeda yang menghasilkan ekspektasi kinerja yang berbeda dan masalah keamanan tambahan. Selain itu, masalah budaya dan politik berpotensi menjadi yang terpenting. Contoh sederhana yang mempengaruhi desain antarmuka pengguna adalah penggunaan warna yang tepat pada formulir (pada layar atau kertas). Budaya yang berbeda menafsirkan warna secara berbeda. Dengan kata lain, dalam lingkungan bisnis multikultural global, menangani masalah budaya lebih dari sekadar memiliki antarmuka pengguna multibahasa. Kita harus mampu menyesuaikan solusi global dengan realitas lokal. Friedman menyebut masalah ini sebagai glokalisasi. Jika tidak, kita hanya akan membuat contoh lain dari proyek pengembangan sistem informasi yang gagal.



Gambar 3-1 Contoh Definisi Persyaratan

Definisi Persyaratan

Laporan definisi persyaratan—biasanya hanya disebut definisi persyaratan—adalah laporan teks langsung yang hanya mencantumkan persyaratan fungsional dan nonfungsional dalam format kerangka. Gambar 3-1 menunjukkan contoh definisi persyaratan untuk sistem janji temu untuk kantor dokter biasa. Perhatikan, itu berisi persyaratan fungsional dan nonfungsional. Persyaratan fungsional termasuk mengatur janji temu, membuat jadwal, dan mencatat ketersediaan dokter individu. Persyaratan nonfungsional mencakup item seperti perkiraan jumlah waktu yang diperlukan untuk melakukan janji temu baru, kebutuhan untuk mendukung pencetakan nirkabel, dan jenis karyawan yang memiliki akses ke berbagai bagian sistem.

Persyaratan diberi nomor dalam format hukum atau garis besar sehingga setiap persyaratan diidentifikasi dengan jelas. Persyaratan pertama dikelompokkan menjadi persyaratan fungsional dan nonfungsional; dalam masing-masing pos tersebut, mereka dikelompokkan lebih lanjut menurut jenis persyaratan nonfungsional atau menurut fungsinya.

Terkadang kebutuhan bisnis diprioritaskan pada definisi kebutuhan. Mereka dapat diberi peringkat dengan kepentingan tinggi, sedang, atau rendah dalam sistem baru, atau mereka dapat diberi label dengan versi sistem yang akan memenuhi persyaratan (misalnya, rilis 1, rilis 2, rilis 3). Praktek ini sangat penting ketika menggunakan metodologi berorientasi objek karena mereka memberikan sistem secara bertahap.

Tujuan paling jelas dari definisi persyaratan adalah untuk menyediakan informasi yang dibutuhkan oleh hasil lain dalam analisis, yang meliputi model fungsional, struktural, dan perilaku, dan untuk mendukung aktivitas dalam desain. Namun, tujuan terpenting dari definisi persyaratan adalah untuk menentukan ruang lingkup sistem. Dokumen tersebut menjelaskan kepada analis apa yang harus dilakukan sistem pada akhirnya. Ketika perbedaan muncul, dokumen berfungsi sebagai tempat untuk pergi untuk klarifikasi.

Menentukan Persyaratan

Menentukan persyaratan untuk definisi persyaratan adalah tugas bisnis dan tugas teknologi informasi. Pada hari-hari awal komputasi, ada anggapan bahwa analis sistem, sebagai ahli dengan sistem komputer, berada dalam posisi terbaik untuk menentukan bagaimana sistem komputer harus beroperasi. Banyak sistem gagal karena tidak memenuhi kebutuhan bisnis pengguna yang sebenarnya. Lambat laun, anggapan itu berubah sehingga pengguna, sebagai pakar bisnis, dipandang sebagai posisi terbaik untuk menentukan bagaimana sistem komputer seharusnya beroperasi. Namun, banyak sistem gagal memberikan manfaat kinerja karena pengguna hanya mengotomatiskan sistem yang tidak efisien yang ada, dan mereka gagal memasukkan peluang baru yang ditawarkan oleh teknologi.

Oleh karena itu, pendekatan yang paling efektif adalah memiliki orang bisnis dan analis yang bekerja sama untuk menentukan kebutuhan bisnis. Namun, terkadang, pengguna tidak tahu persis apa yang mereka inginkan, dan analis perlu membantu mereka menemukan kebutuhan mereka. Serangkaian strategi telah menjadi populer untuk membantu analis melakukan analisis masalah, analisis akar penyebab, analisis durasi, penetapan biaya berbasis aktivitas, pembandingan informal, analisis hasil, analisis teknologi, dan penghapusan aktivitas. Analis dapat menggunakan alat ini ketika mereka perlu memandu pengguna dalam menjelaskan apa yang diinginkan dari suatu sistem. Strategi ini bekerja dengan cara yang sama. Mereka membantu pengguna secara kritis memeriksa keadaan sistem dan proses saat ini (sistem yang ada saat ini), mengidentifikasi dengan tepat apa yang perlu diubah, dan mengembangkan konsep untuk sistem baru (sistem yang akan datang).

Meskipun strategi ini memungkinkan analis untuk membantu pengguna membuat visi untuk sistem baru, mereka tidak cukup untuk mengekstrak informasi tentang persyaratan bisnis terperinci yang diperlukan untuk membangunnya. Oleh karena itu, analis menggunakan portofolio teknik pengumpulan kebutuhan untuk memperoleh informasi dari pengguna. Analis memiliki banyak teknik untuk dipilih: wawancara, kuesioner, observasi, pengembangan aplikasi bersama (JAD), dan analisis dokumen. Informasi yang dikumpulkan menggunakan teknik ini dianalisis secara kritis dan digunakan untuk menyusun laporan definisi persyaratan.

Membuat Definisi Persyaratan

Membuat definisi persyaratan adalah proses berulang dan berkelanjutan dimana analis mengumpulkan informasi dengan teknik pengumpulan persyaratan (misalnya, wawancara,

analisis dokumen), menganalisis informasi secara kritis untuk mengidentifikasi persyaratan bisnis yang sesuai untuk sistem, dan menambahkan persyaratan ke definisi persyaratan laporan. Definisi persyaratan terus diperbarui sehingga tim proyek dan pengguna bisnis dapat merujuknya dan mendapatkan pemahaman yang jelas tentang sistem baru.

Untuk membuat definisi persyaratan, tim proyek pertama-tama menentukan jenis persyaratan fungsional dan nonfungsional yang akan mereka kumpulkan tentang sistem (tentu saja, ini dapat berubah seiring waktu). Ini menjadi bagian utama dari dokumen. Selanjutnya, analis menggunakan berbagai teknik pengumpulan persyaratan untuk mengumpulkan informasi, dan mereka membuat daftar persyaratan bisnis yang diidentifikasi dari informasi tersebut. Akhirnya, analis bekerja dengan seluruh tim proyek dan pengguna bisnis untuk memverifikasi, mengubah, dan melengkapi daftar dan membantu memprioritaskan pentingnya persyaratan yang diidentifikasi.

Proses ini berlanjut selama analisis, dan definisi persyaratan berkembang dari waktu ke waktu saat persyaratan baru diidentifikasi dan saat proyek bergerak ke fase selanjutnya dari Proses Terpadu. Hati-hati: Evolusi definisi persyaratan harus dikelola dengan hati-hati. Tim proyek tidak dapat terus menambahkan definisi persyaratan, atau sistem akan terus tumbuh dan berkembang dan tidak akan pernah selesai. Sebaliknya, tim proyek dengan hati-hati mengidentifikasi persyaratan dan mengevaluasi mana yang sesuai dengan ruang lingkup sistem. Ketika suatu persyaratan mencerminkan kebutuhan bisnis yang nyata tetapi tidak dalam lingkup sistem saat ini atau rilis saat ini, itu akan ditambahkan pada daftar persyaratan masa depan atau diberikan prioritas rendah. Manajemen persyaratan (dan ruang lingkup sistem) adalah salah satu bagian tersulit dalam mengelola proyek.

Masalah Dunia Nyata dengan Penentuan Persyaratan

Avison dan Fitzgerald memberi kami serangkaian masalah yang dapat muncul sehubungan dengan menentukan serangkaian persyaratan yang harus ditangani. Pertama, analis mungkin tidak memiliki akses ke kumpulan pengguna yang benar untuk mengungkap kumpulan persyaratan yang lengkap. Hal ini dapat menyebabkan persyaratan terlewatkan, disalahartikan, dan/atau ditentukan secara berlebihan. Kedua, spesifikasi persyaratan mungkin tidak memadai. Hal ini terutama berlaku dengan teknik ringan yang terkait dengan metodologi tangkas. Ketiga, beberapa persyaratan tidak dapat diketahui pada awal proses pengembangan. Namun, saat sistem dikembangkan, pengguna dan analis akan mendapatkan pemahaman yang lebih baik tentang masalah domain dan teknologi yang berlaku. Hal ini dapat menyebabkan persyaratan fungsional dan nonfungsional baru diidentifikasi dan persyaratan saat ini berkembang atau dibatalkan. Metodologi pengembangan berbasis iteratif dan inkremental, seperti Unified Process dan Agile, dapat membantu dalam kasus ini. Keempat, verifikasi dan validasi persyaratan bisa sangat sulit. Kami mengambil topik ini dalam bab-bab yang berhubungan dengan penciptaan model fungsional (Bab 4), struktural (Bab 5), dan perilaku (Bab 6).

3.4 STRATEGI ANALISIS PERSYARATAN

Sebelum tim proyek dapat menentukan persyaratan apa yang sesuai untuk sistem tertentu, perlu ada visi yang jelas tentang jenis sistem yang akan dibuat dan tingkat perubahan yang akan dibawa ke organisasi. Proses dasar analisis dibagi menjadi tiga langkah: memahami sistem apa adanya, mengidentifikasi perbaikan, dan mengembangkan persyaratan untuk sistem yang akan datang.

Terkadang langkah pertama (yaitu, memahami sistem apa adanya) dilewati atau dilakukan secara sepintas. Ini terjadi ketika tidak ada sistem saat ini, jika sistem dan proses yang ada tidak relevan dengan sistem masa depan, atau jika tim proyek menggunakan RAD

atau metodologi pengembangan tangkas di mana sistem saat in tidak ditekankan. Metodologi RAD, tangkas, dan berorientasi objek yang lebih baru, seperti pengembangan bertahap, pembuatan prototipe, pembuatan prototipe sekali pakai, pemrograman ekstrem, dan Scrum (lihat Bab 1) berfokus hampir secara eksklusif pada peningkatan dan persyaratan sistem yang akan datang, dan mereka menghabiskan sedikit waktu untuk menyelidiki sistem saat in saat ini.

Strategi analisis kebutuhan membantu analis mengarahkan pengguna melalui langkah-langkah analisis sehingga visi sistem dapat dikembangkan. Strategi analisis kebutuhan dan teknik pengumpulan kebutuhan berjalan beriringan. Analis menggunakan teknik pengumpulan kebutuhan untuk mengumpulkan informasi; strategi analisis kebutuhan mendorong jenis informasi yang dikumpulkan dan bagaimana akhirnya dianalisis. Strategi analisis kebutuhan dan pengumpulan kebutuhan terjadi secara bersamaan dan merupakan aktivitas yang saling melengkapi.

Untuk memindahkan pengguna dari sistem saat in ke sistem yang akan datang, seorang analis membutuhkan keterampilan berpikir kritis yang kuat. Berpikir kritis adalah kemampuan untuk mengenali kekuatan dan kelemahan dan menyusun kembali ide dalam bentuk yang lebih baik, dan keterampilan berpikir kritis diperlukan untuk benar-benar memahami masalah dan mengembangkan proses bisnis baru. Keterampilan ini juga diperlukan untuk memeriksa secara menyeluruh hasil pengumpulan persyaratan, untuk mengidentifikasi persyaratan bisnis, dan untuk menerjemahkan persyaratan tersebut ke dalam konsep untuk sistem baru.

Analisa masalah

Teknik analisis kebutuhan yang paling mudah (dan mungkin paling umum digunakan) adalah analisis masalah. Analisis masalah berarti meminta pengguna dan manajer untuk mengidentifikasi masalah dengan sistem saat in dan menjelaskan bagaimana menyelesaikannya dalam sistem yang akan datang. Sebagian besar pengguna memiliki gagasan yang sangat bagus tentang perubahan yang ingin mereka lihat, dan sebagian besar cukup vokal untuk menyarankannya. Sebagian besar perubahan cenderung memecahkan masalah daripada memanfaatkan peluang, tetapi yang terakhir mungkin juga. Perbaikan dari analisis masalah cenderung kecil dan bertahap (misalnya, menyediakan lebih banyak ruang untuk mengetik alamat pelanggan; memberikan laporan baru yang saat ini belum ada).

Jenis perbaikan ini seringkali sangat efektif untuk meningkatkan efisiensi sistem atau kemudahan penggunaan. Namun, sering kali hanya memberikan peningkatan kecil dalam nilai bisnis—sistem baru lebih baik daripada yang lama, tetapi mungkin sulit untuk mengidentifikasi manfaat moneter yang signifikan dari sistem baru.

Analisis Akar Penyebab

Ide-ide yang dihasilkan oleh analisis masalah cenderung menjadi solusi dari masalah. Semua solusi membuat asumsi tentang sifat masalah, asumsi yang mungkin atau mungkin tidak valid. Dalam pengalaman kami, pengguna (dan kebanyakan orang pada umumnya) cenderung dengan cepat melompat ke solusi tanpa sepenuhnya mempertimbangkan sifat masalahnya. Terkadang solusinya tepat, tetapi sering kali solusi tersebut mengatasi gejala masalah, bukan masalah sebenarnya atau akar penyebab itu sendiri

Sebagai contoh, anggaplah sebuah perusahaan memperhatikan bahwa penggunanya melaporkan kehabisan persediaan. Biaya persediaan stock-out bisa sangat signifikan. Dalam hal ini, karena sering terjadi, pelanggan dapat menemukan sumber lain untuk barang yang mereka beli dari perusahaan. Adalah kepentingan perusahaan untuk menentukan penyebab yang mendasari dan tidak hanya memberikan reaksi spontan seperti secara sewenang-wenang

meningkatkan jumlah persediaan yang disimpan. Di dunia bisnis, tantangannya terletak pada mengidentifikasi akar penyebabnya—beberapa masalah dunia nyata yang sederhana. Pengguna biasanya mengusulkan satu set penyebab untuk masalah yang sedang dipertimbangkan. Solusi yang diusulkan pengguna dapat mengatasi gejala atau akar penyebab, tetapi tanpa analisis yang cermat, sulit untuk menentukan mana yang ditangani.

Analisis akar penyebab, oleh karena itu, berfokus pada masalah, bukan solusi. Analis memulai dengan meminta pengguna membuat daftar masalah dengan sistem saat ini dan kemudian memprioritaskan masalah dalam urutan kepentingan. Dimulai dengan yang paling penting, pengguna dan/atau analis kemudian menghasilkan semua kemungkinan akar penyebab masalah. Setiap akar penyebab yang mungkin diselidiki (dimulai dengan yang paling mungkin atau paling mudah untuk diperiksa) sampai akar penyebab yang sebenarnya diidentifikasi. Jika ada akar penyebab yang mungkin diidentifikasi untuk beberapa masalah, itu harus diselidiki terlebih dahulu, karena ada kemungkinan besar mereka adalah akar penyebab sebenarnya yang mempengaruhi masalah gejala. Dalam contoh kita, ada beberapa kemungkinan penyebab utama:

- Pemasok perusahaan mungkin tidak mengirimkan pesanan ke perusahaan pada waktu yang tepat.
- Mungkin ada masalah dengan kontrol inventaris perusahaan.
- Tingkat pemesanan ulang dan kuantitas bisa salah.

Terkadang, menggunakan bagan hierarki untuk mewakili hubungan kausal membantu analisis. Seperti yang ditunjukkan Gambar 3-2, ada banyak kemungkinan akar penyebab yang mendasari penyebab tingkat yang lebih tinggi yang diidentifikasi. Poin utama dalam analisis akar masalah adalah selalu menantang yang sudah jelas.

Analisis Durasi

Analisis durasi memerlukan pemeriksaan rinci tentang jumlah waktu yang diperlukan untuk melakukan setiap proses dalam sistem saat in saat ini. Analis mulai dengan menentukan jumlah total waktu yang dibutuhkan, rata-rata, untuk melakukan serangkaian proses bisnis untuk input yang khas. Mereka kemudian menentukan waktu setiap langkah individu (atau subproses) dalam proses bisnis. Waktu untuk menyelesaikan langkah dasar kemudian dijumlahkan dan dibandingkan dengan total untuk keseluruhan proses. Perbedaan yang signifikan antara keduanya—dan menurut pengalaman kami, waktu total seringkali bisa 10 atau bahkan 100 kali lebih lama dari jumlah bagian—menunjukkan bahwa bagian dari proses ini sangat membutuhkan perbaikan besar-besaran.

Sebagai contoh, misalkan analis sedang mengerjakan sistem hipotek rumah dan menemukan bahwa rata-rata, dibutuhkan tiga puluh hari bagi bank untuk menyetujui hipotek. Mereka kemudian melihat setiap langkah dasar dalam proses (misalnya, entri data, pemeriksaan kredit, pencarian judul, penilaian) dan menemukan bahwa jumlah total waktu yang benar-benar dihabiskan untuk setiap hipotek adalah sekitar delapan jam. Ini adalah indikasi kuat bahwa keseluruhan proses rusak parah, karena butuh tiga puluh hari untuk melakukan pekerjaan satu hari.

Masalah-masalah ini mungkin terjadi karena prosesnya sangat terfragmentasi. Banyak orang yang berbeda harus melakukan aktivitas yang berbeda sebelum proses selesai. Dalam contoh hipotek, aplikasi mungkin duduk di meja banyak orang untuk waktu yang lama sebelum diproses.

Proses di mana banyak orang yang berbeda bekerja pada bagian kecil dari input adalah kandidat utama untuk integrasi atau paralelisasi proses. Integrasi proses berarti mengubah

proses fundamental sehingga lebih sedikit orang yang mengerjakan input, yang seringkali memerlukan perubahan proses dan pelatihan ulang staf untuk melakukan tugas yang lebih luas. Paralelisasi proses berarti mengubah proses sehingga semua langkah individu dilakukan pada waktu yang sama. Misalnya, dalam kasus aplikasi hipotek, mungkin tidak ada alasan bahwa pemeriksaan kredit tidak dapat dilakukan pada saat yang sama dengan pemeriksaan penilaian dan hak milik.

Penetapan Biaya Berbasis Aktivitas

Penetapan biaya berdasarkan aktivitas adalah analisis yang serupa; itu memeriksa biaya setiap proses utama atau langkah dalam proses bisnis daripada waktu yang dibutuhkan. Analis mengidentifikasi biaya yang terkait dengan setiap langkah atau proses fungsional dasar, mengidentifikasi proses yang paling mahal, dan memfokuskan upaya peningkatan mereka pada proses tersebut.

Menetapkan biaya secara konseptual sederhana. Analis hanya memeriksa biaya langsung tenaga kerja dan bahan untuk setiap input. Biaya bahan mudah dibebankan dalam proses manufaktur, sedangkan biaya tenaga kerja biasanya dihitung berdasarkan jumlah waktu yang dihabiskan untuk input dan biaya per jam staf. Namun, seperti yang mungkin Anda ingat dari kursus akuntansi manajerial, ada biaya tidak langsung, seperti sewa, depresiasi, dan sebagainya, yang juga dapat dimasukkan dalam biaya aktivitas.

Pembandingan Informal

Benchmarking mengacu pada mempelajari bagaimana organisasi lain melakukan proses bisnis untuk mempelajari bagaimana organisasi Anda dapat melakukan sesuatu yang lebih baik. Benchmarking membantu organisasi dengan memperkenalkan ide-ide yang mungkin tidak pernah dipertimbangkan oleh karyawan tetapi memiliki potensi untuk menambah nilai.

Pembandingan informal cukup umum untuk proses bisnis yang dihadapi pelanggan (yaitu, proses yang berinteraksi dengan pelanggan). Dengan benchmarking informal, para manajer dan analis berpikir tentang organisasi lain atau mengunjungi mereka sebagai pelanggan untuk melihat bagaimana proses bisnis dilakukan. Dalam banyak kasus, bisnis yang dipelajari mungkin merupakan pemimpin yang dikenal dalam industri atau hanya perusahaan terkait.

Analisis Hasil

Analisis hasil berfokus pada pemahaman hasil mendasar yang memberikan nilai kepada pelanggan. Meskipun hasil ini terdengar seolah-olah mereka harus jelas, mereka sering tidak. Misalnya, pertimbangkan perusahaan asuransi. Salah satu pelanggannya baru saja mengalami kecelakaan mobil. Apa hasil mendasar dari perspektif pelanggan? Secara tradisional, perusahaan asuransi menjawab pertanyaan ini dengan asumsi pelanggan ingin menerima pembayaran asuransi dengan cepat. Namun, bagi pelanggan, pembayaran hanyalah sarana untuk hasil nyata: mobil yang diperbaiki. Perusahaan asuransi mungkin mendapat manfaat dengan memperluas pandangannya tentang proses bisnis melewati batas-batas tradisionalnya untuk memasukkan tidak membayar perbaikan tetapi melakukan perbaikan atau mengontrak bengkel resmi untuk melakukannya.

Dengan pendekatan ini, analis sistem mendorong manajer dan sponsor proyek untuk berpikir dengan hati-hati tentang produk dan layanan organisasi yang memungkinkan untuk

dilakukan oleh pelanggan —dan apa yang dapat mereka lakukan untuk memungkinkan pelanggan melakukannya.

Analisis Teknologi

Banyak perubahan besar dalam bisnis yang dimungkinkan masuknya teknologi baru. Analisis teknologi dimulai dengan meminta para analis dan manajer mengembangkan daftar teknologi yang penting dan menarik. Kemudian kelompok tersebut secara sistematis mengidentifikasi bagaimana setiap teknologi dapat diterapkan pada proses bisnis dan mengidentifikasi seperti apa keuntungan bisnis tersebut. Penting untuk dicatat bahwa analisis teknologi sama sekali tidak menyiratkan adopsi teknologi untuk kepentingan teknologi. Sebaliknya, fokusnya adalah pada penggunaan teknologi baru untuk memenuhi tujuan organisasi.

Penghapusan Aktivitas

Analis dan manajer bekerja sama untuk mengidentifikasi bagaimana organisasi dapat menghilangkan setiap aktivitas dalam proses bisnis, bagaimana fungsi dapat beroperasi tanpanya, dan efek apa yang mungkin terjadi. Awalnya, manajer enggan untuk menyimpulkan bahwa proses dapat dihilangkan. Namun pada akhirnya manajer mencoba menghapus beberapa aktivitas dan melihat perubhana-perubahan yang terjadi selanjutnya.

3.5 PERSYARATAN-TEKNIK PENGUMPULAN

Seorang analis sangat mirip seorang detektif (dan pengguna bisnis terkadang seperti tersangka yang sulit ditangkap). Dia tahu bahwa ada masalah yang harus dipecahkan, sehingga analis harus mencari petunjuk yang mengungkap solusinya. Analis perlu memperhatikan detail, berbicara dengan saksi, dan mengikuti petunjuk seperti yang akan dilakukan Sherlock Holmes. Analis terbaik secara menyeluruh mengumpulkan persyaratan menggunakan berbagai teknik dan memastikan bahwa proses bisnis saat ini dan kebutuhan untuk sistem baru dipahami dengan baik sebelum pindah ke desain. Nantinya, analis tidak ingin mengetahui bahwa mereka memiliki persyaratan utama yang salah karena akan membuat akhir proses pengembangan menyebabkan semua jenis masalah yang tidak diinginkan muncul.

Proses pengumpulan persyaratan digunakan untuk membangun dukungan politik untuk proyek dan membangun kepercayaan serta menjalin hubungan baik antara tim proyek yang membangun sistem dan pengguna yang pada akhirnya akan memilih untuk menggunakan atau tidak menggunakan sistem tersebut. Melibatkan seseorang dalam proses menyiratkan bahwa tim proyek memandang orang itu sebagai sumber daya penting dan menghargai pendapatnya. Semua pemegang kepentingan utama (orang-orang yang dapat mempengaruhi sistem atau yang akan terpengaruh oleh sistem) harus dimasukkan dalam proses pengumpulan kebutuhan. Pemegang kepentingan mungkin termasuk manajer, karyawan, anggota staf, dan bahkan beberapa pelanggan dan pemasok. Jika orang utama tidak terlibat, individu tersebut mungkin merasa diremehkan, yang dapat menyebabkan masalah selama implementasi (misalnya, Bagaimana mereka dapat mengembangkan sistem tanpa masukan saya?).

Tantangan kedua pengumpulan persyaratan adalah memilih cara bagaimana informasi akan dikumpulkan. Ada banyak teknik untuk mengumpulkan persyaratan yang bervariasi dari mengajukan pertanyaan kepada orang-orang, hingga menonton an mengamati mereka dalam bekerja. Pada bagian ini, kami berfokus pada lima teknik yang paling umum digunakan: wawancara, sesi JAD (jenis pertemuan kelompok khusus), kuesioner, analisis dokumen, dan observasi. Setiap teknik memiliki kekuatan dan kelemahannya sendiri, banyak di antaranya saling melengkapi, sehingga sebagian besar proyek menggunakan kombinasi teknik.

Wawancara

Wawancara adalah teknik pengumpulan persyaratan yang paling umum digunakan. Lagi pula, itu wajar—jika Anda perlu mengetahui sesuatu, Anda biasanya bertanya kepada seseorang. Pada umumnya wawancara dilakukan satu lawan satu (satu pewawancara dan satu orang yang diwawancarai), tetapi terkadang karena keterbatasan waktu, beberapa orang diwawancarai pada waktu yang bersamaan. Ada lima langkah dasar untuk proses wawancara:

1) Memilih orang yang diwawancarai
2) Merancang pertanyaan wawancara
3) Mempersiapkan wawancara
4) Melakukan wawancara
5) Tindak lanjut pasca wawancara.

Langkah pertama dalam wawancara adalah membuat daftar jadwal wawancara siapa yang akan diwawancarai, kapan, dan untuk tujuan apa (lihat Gambar 3-3). Jadwal dapat berupa daftar informal yang digunakan untuk membantu mengatur waktu rapat atau daftar formal yang dimasukkan ke dalam rencana kerja. Orang-orang yang muncul dalam jadwal wawancara dipilih berdasarkan kebutuhan informasi analis. Sponsor proyek, pengguna bisnis utama, dan anggota lain dari tim proyek dapat membantu analis menentukan siapa dalam organisasi yang paling dapat memberikan informasi penting tentang persyaratan. Orang-orang ini terdaftar pada jadwal wawancara dalam urutan di mana mereka harus diwawancarai.

Orang-orang di berbagai tingkat organisasi memiliki perspektif yang berbeda-beda tentang sistem, jadi penting untuk menyertakan manajer yang mengelola proses dan staf yang benar-benar melakukan proses untuk mendapatkan perspektif tingkat tinggi dan tingkat rendah tentang suatu masalah. Juga, jenis subjek wawancara yang dibutuhkan dapat berubah seiring waktu. Misalnya, pada awal proyek, analis memiliki pemahaman yang terbatas tentang proses bisnis apa adanya. Untuk memulai dengan mewawancarai satu atau dua manajer senior untuk mendapatkan pandangan strategis dan kemudian beralih ke manajer tingkat menengah yang dapat memberikan informasi yang luas dan menyeluruh tentang proses bisnis dan peran yang diharapkan dari sistem yang sedang dikembangkan merupakan hal yang umum. Setelah analis memiliki pemahaman yang baik tentang gambaran besar, manajer tingkat bawah dan anggota staf dapat mengisi rincian yang tepat tentang cara kerja proses. Seperti kebanyakan hal lain tentang analisis sistem, ini adalah proses berulang-mulai dengan manajer senior, pindah ke manajer tingkat menengah, kemudian anggota staf, kembali ke manajer tingkat menengah, dan dan selanjutnya, tergantung pada informasi apa yang dibutuhkan di sepanjang proses.

Daftar orang yang diwawancarai biasanya bertambah, sering kali sebesar 50 hingga 75 persen. Saat orang diwawancarai, lebih banyak informasi yang dibutuhkan dan orang tambahan yang dapat memberikan informasi mungkin akan diidentifikasi.

Name	Position	Purpose of Interview	Meeting
Andria McClellan	Director, Accounting	Strategic vision for new accounting system	Mon., March 1 8:00–10:00 AM
Jennifer Draper	Manager, Accounts Receivable	Current problems with accounts receivable process; future goals	Mon., March 1 2:00–3:15 PM
Mark Goodin	Manager, Accounts Payable	Current problems with accounts payable process; future goals	Mon., March 1 4:00–5:15 PM
Anne Asher	Supervisor, Data Entry	Accounts receivable and payable processes	Wed., March 3 10:00–11:00 AM
Fernando Merce	Data Entry Clerk	Accounts receivable and payable processes	Wed., March 3 1:00–3:00 PM

Gambar 3-3 Contoh Jadwal Wawancara

Ada tiga jenis pertanyaan wawancara: pertanyaan tertutup, pertanyaan terbuka, dan pertanyaan penyelidikan. Pertanyaan tertutup adalah pertanyaan yang membutuhkan jawaban spesifik. Mereka mirip dengan pertanyaan pilihan ganda atau aritmatika pada ujian (lihat Gambar 3-4). Pertanyaan tertutup digunakan ketika seorang analis mencari informasi yang spesifik dan tepat (misalnya, berapa banyak permintaan kartu kredit yang diterima per hari). Secara umum, pertanyaan yang tepat adalah yang terbaik. Misalnya, daripada bertanya, Apakah Anda menangani banyak permintaan? lebih baik bertanya, Berapa banyak permintaan yang Anda proses per hari? Pertanyaan tertutup memungkinkan analis untuk mengontrol wawancara dan memperoleh informasi yang mereka butuhkan. Namun, jenis pertanyaan ini tidak mengungkap mengapa jawabannya seperti itu, juga tidak mengungkap informasi yang menurut pewawancara tidak akan ditanyakan sebelumnya.

Pertanyaan terbuka adalah mereka yang meninggalkan ruang untuk elaborasi pada bagian dari orang yang diwawancarai. Mereka serupa dalam banyak hal dengan pertanyaan esai yang mungkin Anda temukan dalam ujian (lihat Gambar 3-4 untuk contoh). Pertanyaan terbuka dirancang untuk mengumpulkan informasi yang kaya dan memberi orang yang diwawancarai lebih banyak kendali atas informasi yang terungkap selama wawancara. Kadang-kadang informasi yang dipilih oleh orang yang diwawancarai untuk didiskusikan mengungkap informasi yang sama pentingnya dengan jawabannya (misalnya, jika orang yang diwawancarai hanya berbicara tentang departemen lain ketika ditanya masalah, ini mungkin menunjukkan bahwa dia enggan untuk mengakui masalahnya sendiri. masalah).

Jenis pertanyaan ketiga adalah pertanyaan penyelidikan. Pertanyaan penyelidikan menindaklanjuti apa yang baru saja dibahas untuk mempelajari lebih lanjut, ini sering digunakan ketika pewawancara tidak jelas tentang jawaban orang yang diwawancarai. Mereka mendorong orang yang diwawancarai untuk memperluas atau mengkonfirmasi informasi dari tanggapan sebelumnya, dan mereka memberi isyarat bahwa pewawancara mendengarkan dan tertarik pada topik yang sedang dibahas. Banyak analis pemula enggan untuk menggunakan pertanyaan penyelidikan karena mereka takut orang yang diwawancarai mungkin akan ditentang atau karena mereka percaya itu menunjukkan bahwa mereka tidak mengerti apa yang dikatakan orang yang diwawancarai. Ketika dilakukan dengan sopan, pertanyaan penyelidikan bisa menjadi alat yang ampuh dalam pengumpulan kebutuhan.

Secara umum, pewawancara tidak boleh mengajukan pertanyaan tentang informasi yang tersedia dari sumber lain. Misalnya, daripada menanyakan informasi apa yang digunakan untuk melakukan suatu tugas, lebih mudah untuk menunjukkan formulir atau laporan kepada orang yang diwawancarai (lihat bagian tentang analisis dokumen) dan menanyakan informasi apa yang digunakan. Ini membantu memfokuskan orang yang diwawancarai dan menghemat waktu, karena orang yang diwawancarai tidak perlu menjelaskan detail informasi—dia hanya perlu menunjukkannya pada formulir atau laporan.

Tidak ada jenis pertanyaan yang lebih baik dari yang lain, dan kombinasi pertanyaan biasanya digunakan selama wawancara. Pada tahap awal proyek pengembangan SI, proses apa adanya dapat menjadi tidak jelas, sehingga proses wawancara dimulai dengan wawancara tidak terstruktur, wawancara yang mencari informasi yang luas dan didefinisikan secara kasar. Dalam hal ini, pewawancara memiliki pengertian umum tentang informasi yang dibutuhkan tetapi hanya memiliki sedikit pertanyaan tertutup untuk diajukan. Ini adalah wawancara yang paling menantang untuk dilakukan karena mengharuskan pewawancara untuk mengajukan pertanyaan terbuka dan menyelidiki informasi penting dengan cepat.

Seiring berjalannya proyek, analis menjadi lebih memahami proses bisnis dan membutuhkan informasi yang sangat spesifik tentang bagaimana proses bisnis dilakukan (misalnya, bagaimana tepatnya kartu kredit pelanggan disetujui). Pada saat ini, analis melakukan wawancara terstruktur, di mana serangkaian pertanyaan spesifik dikembangkan sebelum wawancara. Biasanya ada lebih banyak pertanyaan tertutup dalam wawancara terstruktur daripada dalam pendekatan tidak terstruktur.

Apapun jenis wawancara yang dilakukan, pertanyaan wawancara harus disusun dalam urutan yang logis agar wawancara mengalir dengan baik. Misalnya, ketika mencoba mengumpulkan informasi tentang proses bisnis saat ini, akan berguna untuk bergerak dalam urutan yang logis melalui proses atau dari masalah yang paling penting ke yang paling tidak penting.

Ada dua pendekatan mendasar untuk mengatur pertanyaan wawancara: top down atau bottom up (lihat Gambar 3-5). Dengan wawancara top-down, pewawancara memulai dengan masalah umum yang luas dan secara bertahap bekerja ke arah yang lebih spesifik. Dengan wawancara bottom-up, pewawancara memulai dengan pertanyaan yang sangat spesifik dan beralih ke pertanyaan yang luas. Dalam praktiknya, analis mencampurkan dua pendekatan, dimulai dengan masalah umum yang luas, pindah ke pertanyaan khusus, dan kemudian kembali ke masalah umum.

Pendekatan top-down adalah strategi yang tepat untuk sebagian besar wawancara (ini tentu saja merupakan pendekatan yang paling umum). Pendekatan top-down memungkinkan orang yang diwawancarai menjadi terbiasa dengan topik sebelum dia perlu memberikan spesifik. Hal ini juga memungkinkan pewawancara untuk memahami masalah sebelum pindah ke rincian karena pewawancara mungkin tidak memiliki informasi yang cukup pada awal wawancara untuk mengajukan pertanyaan yang sangat spesifik. Mungkin yang paling penting, pendekatan top-down memungkinkan orang yang diwawancarai untuk mengangkat serangkaian masalah gambaran besar sebelum menjadi terjerat dalam detail, sehingga pewawancara cenderung tidak melewatkan masalah penting.

Satu kasus di mana strategi bottom-up mungkin lebih disukai adalah ketika analis telah mengumpulkan banyak informasi tentang masalah dan hanya perlu mengisi beberapa lubang dengan detail. Wawancara bottom-up mungkin tepat jika anggota staf tingkat bawah merasa terancam atau tidak mampu menjawab pertanyaan tingkat tinggi. Misalnya, Bagaimana kita dapat meningkatkan layanan pelanggan? mungkin pertanyaan yang terlalu luas untuk petugas layanan pelanggan, sedangkan pertanyaan spesifik mudah dijawab (misalnya, Bagaimana kita

bisa mempercepat pengembalian pelanggan?). Bagaimanapun, semua wawancara harus dimulai dengan pertanyaan yang tidak kontroversial dan kemudian secara bertahap beralih ke masalah yang lebih kontroversial setelah pewawancara mengembangkan beberapa hubungan dengan orang yang diwawancarai.

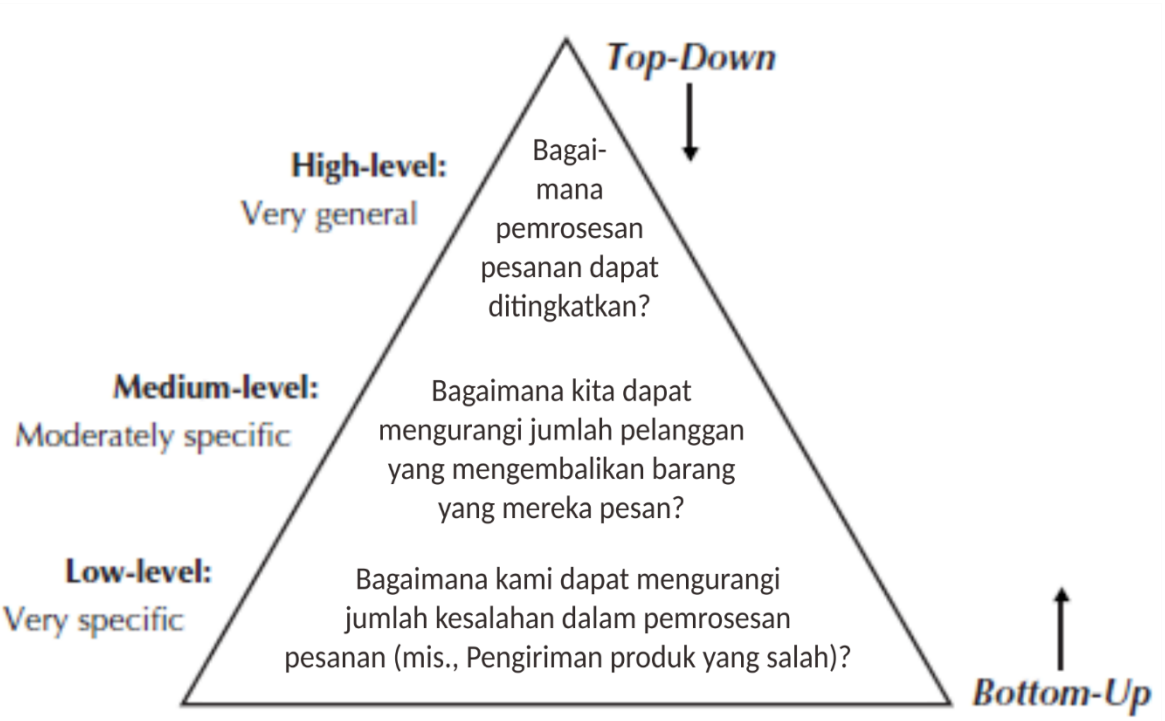
Jenis Pertanyaan	**Contoh**
Pertanyaan Tertutup	• Berapa banyak pesanan telepon yang diterima per hari? • Bagaimana cara pelanggan melakukan pemesanan? • Informasi apa yang hilang dari laporan penjualan bulanan?
Pertanyaan Terbuka	• Apa yang Anda pikirkan tentang mesin yang tersedia? • Apa saja masalah yang Anda hadapi sehari-hari? • Apa saja peningkatan yang ingin Anda lihat dalam sistem baru?
Pertanyaan Penyelidikan	• Mengapa? • Dapatkah Anda memberi Saya contoh? • Dapatkah Anda menjelaskan lebih detail lagi?

Gambar 3-4 Tiga Jenis Soal

Penting untuk mempersiapkan wawancara dengan cara yang sama seperti Anda mempersiapkan diri untuk memberikan presentasi. Pewawancara harus memiliki rencana wawancara umum yang mencantumkan pertanyaan yang akan diajukan dalam urutan yang sesuai, harus mengantisipasi kemungkinan jawaban dan memberikan tindak lanjut dengan mereka, dan harus mengidentifikasi segmen antara topik terkait. Pewawancara harus mengkonfirmasi area di mana orang yang diwawancarai memiliki pengetahuan agar tidak mengajukan pertanyaan yang tidak dapat dijawab oleh orang yang diwawancarai. Tinjau area topik, pertanyaan, dan rencana wawancara, dan putuskan dengan jelas mana yang memiliki prioritas terbesar jika waktu hampir habis.

Secara umum, wawancara terstruktur dengan pertanyaan tertutup membutuhkan lebih banyak waktu untuk persiapan daripada wawancara tidak terstruktur. Beberapa analis pemula lebih suka wawancara tidak terstruktur, berpikir bahwa mereka dapat melakukannya. Ini sangat berbahaya dan seringkali kontraproduktif, karena informasi apa pun yang tidak dikumpulkan dalam wawancara pertama akan memerlukan upaya tindak lanjut, dan sebagian besar pengguna tidak suka diwawancarai berulang kali tentang masalah yang sama.

Pewawancara harus memastikan untuk mempersiapkan orang yang diwawancarai juga. Ketika wawancara dijadwalkan, orang yang diwawancarai harus diberi tahu terlebih dahulu apa alasan wawancara dan area mana saja yang akan dibahas sehingga dia punya waktu untuk memikirkan masalah dan mengatur pikirannya. Hal ini sangat penting ketika pewawancara adalah orang luar bagi organisasi dan untuk karyawan tingkat bawah, yang sering tidak dimintai pendapatnya dan yang mungkin tidak yakin tentang mengapa mereka diwawancarai.



Gambar 3-5 Strategi Bertanya Top-Down dan Bottom-Up

Tujuan pertama adalah membangun hubungan baik dengan orang yang diwawancarai, sehingga dia mempercayai pewawancara dan mau mengatakan yang sebenarnya, tidak hanya memberikan jawaban yang dia pikir diinginkan. Pewawancara harus tampak sebagai pencari informasi yang profesional dan tidak memihak. Wawancara harus dimulai dengan penjelasan mengapa pewawancara ada di sana dan mengapa dia memilih untuk mewawancarai orang tersebut; maka pewawancara harus pindah ke pertanyaan wawancara yang direncanakan.

Sangat penting untuk mencatat semua informasi yang diberikan oleh orang yang diwawancarai dengan hati-hati. Dalam pengalaman kami, pendekatan terbaik adalah membuat catatan yang cermat—tuliskan semua yang dikatakan orang yang diwawancarai, bahkan jika itu tidak segera tampak relevan. Pewawancara tidak perlu takut untuk meminta orang tersebut untuk memperlambat atau berhenti saat menulis, karena ini adalah indikasi yang jelas bahwa informasi yang diwawancarai itu penting. Salah satu isu yang berpotensi kontroversial adalah apakah wawancara akan direkam atau tidak. Rekaman memastikan bahwa pewawancara tidak melewatkan poin penting, tetapi dapat mengintimidasi orang yang diwawancarai. Sebagian besar organisasi memiliki kebijakan atau praktik yang diterima secara umum tentang perekaman wawancara, sehingga harus ditentukan sebelum wawancara. Jika pewawancara khawatir tentang informasi yang hilang dan tidak dapat merekam wawancara, maka dia dapat membawa orang kedua untuk membuat catatan terperinci.

Saat wawancara berlangsung, penting untuk memahami masalah yang dibahas. Jika pewawancara tidak mengerti sesuatu, dia harus meminta klarifikasi. Pewawancara tidak perlu takut untuk mengajukan pertanyaan bodoh, karena satu-satunya hal yang lebih buruk daripada terlihat bodoh adalah menjadi bodoh dengan tidak memahami sesuatu. Jika pewawancara tidak memahami sesuatu selama wawancara, dia pasti tidak akan memahaminya setelahnya. Jargon harus diakui dan didefinisikan; jargon apa pun yang tidak dipahami harus diklarifikasi. Salah satu strategi yang baik untuk meningkatkan pemahaman selama wawancara adalah secara berkala merangkum poin-poin utama yang dikomunikasikan oleh orang yang diwawancarai. Ini menghindari kesalahpahaman dan juga menunjukkan bahwa pewawancara mendengarkan.

Terakhir, fakta harus dipisahkan dari opini. Orang yang diwawancarai mungkin mengatakan, misalnya, Kami memproses terlalu banyak permintaan kartu kredit. Ini adalah opini, dan berguna untuk menindaklanjutinya dengan pertanyaan penyelidikan yang meminta dukungan untuk pernyataan tersebut (misalnya, Oh, berapa banyak yang Anda proses dalam sehari?). Sangat membantu untuk memeriksa fakta karena perbedaan antara fakta dan pendapat orang yang diwawancarai dapat menunjukkan area utama untuk perbaikan. Misalkan orang yang diwawancarai mengeluh tentang jumlah kesalahan yang tinggi atau meningkat, tetapi log menunjukkan bahwa kesalahan telah berkurang. Hal ini menunjukkan bahwa kesalahan dipandang sebagai masalah yang sangat penting yang harus ditangani oleh sistem baru, bahkan jika mereka menurun.

Saat wawancara hampir berakhir, orang yang diwawancarai harus memiliki waktu untuk mengajukan pertanyaan atau memberikan informasi yang menurutnya penting tetapi bukan bagian dari rencana wawancara. Dalam kebanyakan kasus, orang yang diwawancarai tidak memiliki kekhawatiran atau informasi tambahan, tetapi dalam beberapa kasus ini mengarah pada informasi yang tidak terduga, tetapi penting. Demikian juga, akan berguna untuk menanyakan orang yang diwawancarai apakah ada orang lain yang harus diwawancarai. Wawancara harus berakhir tepat waktu (jika perlu, beberapa topik dapat dihilangkan atau wawancara lain dapat dijadwalkan).

Sebagai langkah terakhir dalam wawancara, pewawancara harus menjelaskan secara singkat apa yang akan terjadi. Pewawancara tidak boleh terlalu dini menjanjikan fitur-fitur tertentu dalam sistem baru atau tanggal pengiriman tertentu, tetapi dia harus meyakinkan orang yang diwawancarai bahwa waktunya dihabiskan dengan baik dan sangat membantu proyek.

Setelah wawancara selesai, analis perlu menyiapkan laporan wawancara yang menjelaskan informasi dari wawancara tersebut (Gambar 3-6). Laporan tersebut berisi catatan wawancara, informasi yang dikumpulkan selama wawancara dan diringkas dalam format yang berguna. Secara umum, laporan wawancara harus ditulis dalam waktu empat puluh delapan jam setelah wawancara, karena semakin lama pewawancara menunggu, semakin besar kemungkinan dia akan melupakan informasi.

Seringkali, laporan wawancara dikirim ke orang yang diwawancarai dengan permintaan untuk membacanya dan memberi tahu analis tentang klarifikasi atau pembaruan. Orang yang diwawancarai perlu diyakinkan bahwa pewawancara benar-benar menginginkan koreksinya terhadap laporan tersebut. Biasanya ada sedikit perubahan, tetapi kebutuhan akan perubahan signifikan menunjukkan bahwa wawancara kedua akan diperlukan. Jangan pernah mendistribusikan informasi seseorang tanpa persetujuan sebelumnya.

Pengembangan Aplikasi Bersama/*Joint Application Development* (JAD)

JAD adalah teknik pengumpulan informasi yang memungkinkan tim proyek, pengguna, dan manajemen bekerja sama untuk mengidentifikasi persyaratan untuk sistem. IBM mengembangkan teknik JAD pada akhir 1970-an, dan sering kali merupakan metode yang paling berguna untuk mengumpulkan informasi dari pengguna.

Orang yang Diwawancarai: Abigail **Direktur, Sumber Daya Manusia** **Pewawancara: Agus Wibowo** **Tujuan Wawancara:** • Memahami laporan yang dihasilkan untuk Sumber Daya Manusia oleh sistem saat ini

- Tentukan kebutuhan informasi untuk sistem masa depan

Ringkasan Wawancara:

- Contoh laporan dari semua laporan HR saat ini terlampir pada laporan ini. Informasi yang tidak digunakan dan informasi yang hilang dicatat pada laporan.
- Dua masalah terbesar dengan sistem saat ini adalah:
    1. Data terlalu lama (Departemen SDM membutuhkan informasi akhir bulan; saat ini, informasi diberikan kepada mereka setelah penundaan tiga minggu)
    2. Data berkualitas buruk (seringkali laporan harus direkonsiliasi dengan database SDM departemen)
- Kesalahan data yang paling umum ditemukan dalam sistem saat ini termasuk informasi tingkat pekerjaan yang salah dan informasi gaji yang hilang.

Item:

- Dapatkan laporan daftar karyawan saat ini dari Mary Skudrna (ekstensi 4355).
- Verifikasi perhitungan yang digunakan untuk menentukan waktu liburan bersama Mary Skudrna.
- Jadwalkan wawancara dengan Jim Wack (ekstensi 2337) mengenai penyebab masalah kualitas data.

Catatan Detail: Lihat transkrip terlampir.

Gambar 3-6 Laporan Wawancara

Capers Jones mengklaim bahwa JAD dapat mengurangi scope creep hingga 50 persen dan mencegah persyaratan sistem menjadi terlalu spesifik atau terlalu kabur, yang keduanya menyebabkan masalah selama tahap selanjutnya dari proses pengembangan.

JAD adalah proses terstruktur di mana sepuluh hingga dua puluh pengguna bertemu bersama di bawah arahan seorang fasilitator yang ahli dalam teknik JAD. Fasilitator menetapkan agenda pertemuan dan memandu diskusi tetapi tidak ikut dalam diskusi sebagai peserta. Dia tidak memberikan ide atau pendapat tentang topik yang sedang dibahas agar tetap netral selama sesi. Fasilitator harus ahli baik dalam teknik proses kelompok maupun teknik analisis dan desain sistem. Satu atau dua juru tulis membantu fasilitator dengan mencatat, membuat salinan, dan segera. Seringkali juru tulis menggunakan komputer dan CASE-tool untuk merekam informasi sebagai proses sesi JAD.

Kelompok JAD bertemu selama beberapa jam, beberapa hari, atau beberapa minggu sampai semua masalah telah dibahas dan informasi yang dibutuhkan dikumpulkan. Sebagian besar sesi JAD berlangsung di ruang pertemuan yang disiapkan khusus, jauh dari kantor peserta sehingga tidak terganggu. Ruang pertemuan biasanya diatur dalam bentuk U sehingga semua peserta dapat dengan mudah melihat satu sama lain. Di depan ruangan (bagian terbuka dari U), terdapat papan tulis, flip chart, dan/atau proyektor untuk digunakan oleh fasilitator yang memimpin diskusi.

Mengembangkan Keterampilan Interpersonal

Keterampilan interpersonal adalah keterampilan yang memungkinkan Anda mengembangkan hubungan baik dengan orang lain, dan keterampilan ini sangat penting untuk wawancara. Mereka membantu Anda berkomunikasi dengan orang lain secara efektif. Beberapa orang mengembangkan keterampilan interpersonal yang baik pada usia dini; mereka tampaknya hanya tahu bagaimana berkomunikasi dan berinteraksi dengan orang lain. Orang lain kurang beruntung dan perlu bekerja keras untuk mengembangkan keterampilan mereka.

Keterampilan interpersonal, seperti kebanyakan keterampilan, dapat dipelajari. Berikut beberapa tipsnya:

- **Jangan khawatir, berbahagialah.** Orang yang bahagia memancarkan kepercayaan diri dan memproyeksikan perasaan mereka pada orang lain. Cobalah mewawancarai seseorang sambil tersenyum dan kemudian mewawancarai orang lain sambil mengerutkan kening dan lihat apa yang terjadi.
- **Perhatian**. Perhatikan apa yang dikatakan orang lain (yang lebih sulit dari yang Anda kira). Lihat berapa kali Anda mendapati diri Anda memikirkan sesuatu selain percakapan yang ada.
- **Ringkas poin-poin penting**. Di akhir setiap tema atau ide utama yang dijelaskan seseorang, ulangi poin-poin utama kembali ke pembicara (misalnya, Biarkan saya memastikan saya mengerti. Isu utamanya adalah . . .). Ini menunjukkan bahwa Anda menganggap informasi itu penting, dan juga memaksa Anda untuk memperhatikan (Anda tidak dapat mengulangi apa yang tidak Anda dengar).
- **Singkat.** Saat Anda berbicara, ringkaslah. Tujuan dalam wawancara (dan dalam sebagian besar kehidupan) adalah untuk belajar, bukan untuk mengesankan. Semakin banyak Anda berbicara, semakin sedikit waktu yang Anda berikan kepada orang lain.
- **Jujur**. Jawab semua pertanyaan dengan jujur, dan jika Anda tidak tahu jawabannya, katakan saja.
- **Perhatikan bahasa tubuh (milik Anda dan mereka).** Cara seseorang duduk atau berdiri menyampaikan banyak informasi. Secara umum, seseorang yang tertarik dengan apa yang Anda katakan duduk atau mencondongkan tubuh ke depan, melakukan kontak mata, dan sering menyentuh wajahnya. Seseorang yang bersandar jauh dari Anda atau dengan lengan di belakang kursi tidak tertarik. Lengan yang disilangkan menunjukkan sikap defensif atau ketidakpastian, dan curam (duduk dengan tangan terangkat di depan tubuh dengan ujung jari menyentuh) menunjukkan perasaan superioritas.

JAD menderita masalah tradisional yang terkait dengan kelompok: Kadang-kadang orang enggan untuk menantang pendapat orang lain (terutama bos mereka), beberapa orang sering mendominasi diskusi, dan tidak semua orang berpartisipasi. Dalam kelompok yang beranggotakan lima belas orang, misalnya, jika setiap orang berpartisipasi secara setara, maka setiap orang hanya dapat berbicara selama empat menit setiap jam dan harus mendengarkan selama lima puluh enam menit yang tersisa—bukan cara yang sangat efisien untuk mengumpulkan informasi.

Sebuah bentuk baru JAD yang disebut electronic JAD, atau e-JAD, mencoba untuk mengatasi masalah ini dengan menggunakan groupware. Di ruang pertemuan e-JAD, setiap peserta menggunakan perangkat lunak khusus pada komputer jaringan untuk mengirim ide dan pendapat anonim kepada orang lain. Dengan cara ini, semua peserta dapat berkontribusi pada saat yang sama tanpa takut akan pembalasan dari orang-orang yang berbeda pendapat. Penelitian awal menunjukkan bahwa e-JAD dapat mengurangi waktu yang dibutuhkan untuk menjalankan sesi JAD sebesar 50 hingga 80 persen. Pendekatan JAD yang baik mengikuti serangkaian lima langkah.

Peserta JAD dipilih dengan cara yang sama seperti peserta wawancara, berdasarkan informasi yang dapat mereka sumbangkan untuk memberikan campuran yang luas dari tingkat organisasi dan untuk membangun dukungan politik untuk sistem baru. Kebutuhan semua peserta JAD untuk berada jauh dari kantor mereka pada saat yang sama dapat menjadi masalah

besar. Kantor mungkin perlu ditutup atau beroperasi dengan staf kerangka sampai sesi JAD selesai.

Idealnya, peserta yang dibebaskan dari tugas reguler untuk menghadiri sesi JAD harus menjadi orang-orang terbaik di unit bisnis itu. Namun, tanpa dukungan manajemen yang kuat, sesi JAD dapat gagal karena mereka yang dipilih untuk menghadiri sesi JAD adalah orang-orang yang kemungkinannya kecil untuk dilewatkan (yaitu, orang yang paling tidak kompeten).

Fasilitator harus seseorang yang ahli dalam teknik JAD atau e-JAD dan, idealnya, seseorang yang berpengalaman dengan bisnis yang sedang dibahas. Dalam banyak kasus, fasilitator JAD adalah konsultan di luar organisasi karena organisasi mungkin tidak memiliki kebutuhan berulang akan keahlian JAD atau e-JAD. Mengembangkan dan mempertahankan keahlian in-house ini bisa mahal.

Sesi JAD dapat berjalan dari setengah hari hingga beberapa minggu, tergantung pada ukuran dan ruang lingkup proyek. Dalam pengalaman kami, sebagian besar sesi JAD cenderung berlangsung lima hingga sepuluh hari, tersebar dalam periode tiga minggu. Sebagian besar sesi e-JAD cenderung berlangsung satu hingga empat hari dalam periode satu minggu. Sesi JAD dan e-JAD biasanya lebih dari sekadar mengumpulkan informasi dan beralih ke analisis. Misalnya, pengguna dan analis secara kolektif dapat membuat hasil analisis, seperti model fungsional atau definisi persyaratan.

Sesi JAD biasanya dirancang dan terstruktur dengan menggunakan prinsip yang sama seperti wawancara. Sebagian besar sesi JAD dirancang untuk mengumpulkan informasi spesifik dari pengguna, dan ini memerlukan pengembangan serangkaian pertanyaan sebelum rapat. Satu perbedaan antara JAD dan wawancara adalah bahwa semua sesi JAD terstruktur—mereka harus direncanakan dengan hati-hati. Secara umum, pertanyaan tertutup jarang digunakan karena tidak memicu diskusi terbuka dan jujur yang merupakan ciri khas JAD. Dalam pengalaman kami, lebih baik untuk melanjutkan dari atas ke bawah dalam sesi JAD saat mengumpulkan informasi. Biasanya tiga puluh menit dialokasikan untuk setiap item agenda yang terpisah, dan istirahat yang sering dijadwalkan sepanjang hari karena peserta mudah lelah.

Seperti halnya wawancara, penting untuk mempersiapkan analis dan peserta untuk sesi JAD. Karena sesi dapat melampaui kedalaman wawancara biasa dan biasanya dilakukan di luar lokasi, peserta mungkin lebih peduli tentang bagaimana mempersiapkannya. Adalah penting bahwa para peserta memahami apa yang diharapkan dari mereka. Jika tujuan sesi JAD, misalnya, adalah untuk mengembangkan pemahaman tentang sistem saat ini, maka peserta dapat membawa manual prosedur dan dokumen. Jika tujuannya adalah untuk mengidentifikasi perbaikan sistem, maka sebelum mereka datang ke sesi JAD mereka dapat memikirkan bagaimana mereka akan meningkatkan sistem.

Sebagian besar sesi JAD mengikuti agenda formal, dan sebagian besar memiliki aturan dasar formal yang mendefinisikan perilaku yang sesuai. Aturan dasar yang umum termasuk mengikuti jadwal, menghormati pendapat orang lain, menerima ketidaksepakatan, dan memastikan bahwa hanya satu orang yang berbicara pada satu waktu.

Peran fasilitator JAD bisa jadi menantang. Banyak peserta datang ke sesi JAD dengan perasaan yang kuat tentang sistem yang akan dibahas. Menyalurkan perasaan ini sehingga sesi bergerak maju ke arah yang positif dan membuat peserta mengenali dan menerima—tetapi tidak harus menyetujui—pendapat dan situasi yang berbeda dari mereka sendiri membutuhkan keahlian yang signifikan dalam analisis dan desain sistem, JAD, dan keterampilan interpersonal. Beberapa analis sistem berusaha untuk memfasilitasi sesi JAD tanpa dilatih

dalam teknik JAD, dan sebagian besar magang dengan fasilitator JAD yang terampil sebelum mereka mencoba untuk memimpin sesi pertama mereka.

Fasilitator JAD melakukan tiga fungsi utama. Pertama, dia memastikan bahwa kelompoknya tetap pada agenda. Satu-satunya alasan untuk menyimpang dari agenda adalah ketika menjadi jelas bagi fasilitator, pemimpin proyek, dan sponsor proyek bahwa sesi JAD telah menghasilkan beberapa informasi baru yang tidak terduga dan membutuhkan sesi JAD (dan mungkin proyek) untuk bergerak secara bertahap. arah baru. Ketika peserta berusaha mengalihkan diskusi dari agenda, fasilitator harus tegas tetapi sopan dalam mengarahkan diskusi kembali ke agenda dan mengembalikan kelompok ke jalurnya.

Kedua, fasilitator harus membantu kelompok memahami istilah teknis dan jargon yang melingkupi proses pengembangan sistem dan membantu peserta memahami teknik analisis khusus yang digunakan. Peserta adalah ahli di bidangnya, atau bagian bisnis mereka, tetapi mereka bukan ahli dalam analisis sistem. Fasilitator harus, oleh karena itu, meminimalkan pembelajaran yang diperlukan dan mengajar peserta bagaimana memberikan informasi yang benar secara efektif.

Ketiga, fasilitator mencatat masukan kelompok pada area pajangan publik, yang dapat berupa papan tulis, flip chart, atau pajangan komputer. Dia menyusun informasi yang diberikan kelompok dan membantu kelompok mengenali isu-isu utama dan solusi penting. Fasilitator harus tetap netral setiap saat dan hanya membantu kelompok melalui proses tersebut. Saat fasilitator memberikan pendapat tentang suatu masalah, kelompok akan melihat dia bukan sebagai pihak yang netral melainkan sebagai seseorang yang dapat mencoba untuk mempengaruhi kelompok ke dalam beberapa solusi yang telah ditentukan.

Namun, ini tidak berarti bahwa fasilitator tidak boleh mencoba membantu kelompok menyelesaikan masalah. Misalnya, jika dua item tampak sama bagi fasilitator, fasilitator tidak boleh mengatakan, "Saya pikir ini mungkin serupa." Sebaliknya, fasilitator harus bertanya, "Apakah ini serupa?" Jika kelompok memutuskan demikian, fasilitator dapat menggabungkan mereka dan melanjutkan. Namun, jika kelompok memutuskan bahwa mereka tidak serupa (terlepas dari apa yang diyakini fasilitator), fasilitator harus menerima keputusan tersebut dan melanjutkan. Kelompok selalu benar, dan fasilitator tidak memiliki pendapat.

Seperti halnya wawancara, laporan pasca-sesi JAD disiapkan dan diedarkan di antara peserta sesi. Laporan pasca-sesi pada dasarnya sama dengan laporan wawancara pada Gambar 3-6. Karena sesi JAD lebih lama dan memberikan lebih banyak informasi, biasanya diperlukan waktu satu atau dua minggu setelah sesi JAD sebelum laporan selesai.

Kuesioner

Kuesioner adalah serangkaian pertanyaan tertulis yang digunakan untuk memperoleh informasi dari individu. Kuesioner sering digunakan ketika ada banyak orang yang membutuhkan informasi dan pendapat. Dalam pengalaman kami, kuesioner adalah teknik umum dengan sistem yang dimaksudkan untuk digunakan di luar organisasi (misalnya, oleh pelanggan atau vendor) atau untuk sistem dengan pengguna bisnis yang tersebar di banyak lokasi geografis. Kebanyakan orang secara otomatis memikirkan kertas ketika mereka memikirkan kuesioner, tetapi hari ini lebih banyak kuesioner didistribusikan dalam bentuk elektronik, baik melalui email atau di Web. Distribusi elektronik dapat menghemat jumlah uang yang signifikan dibandingkan dengan menyebarkan kuesioner kertas. Proses yang baik untuk digunakan saat menggunakan kuesioner mengikuti empat langkah.

Seperti halnya wawancara dan sesi JAD, langkah pertama adalah mengidentifikasi individu yang akan dikirimi kuesioner. Namun, tidak biasa untuk memilih setiap orang yang

dapat memberikan informasi yang berguna. Pendekatan standar adalah memilih sampel, atau subset, dari orang-orang yang mewakili seluruh kelompok. Pedoman pengambilan sampel dibahas di sebagian besar buku statistik, dan sebagian besar sekolah bisnis menyertakan kursus yang mencakup topik tersebut, jadi kami tidak membahasnya di sini. Poin penting dalam memilih sampel, bagaimanapun, adalah menyadari bahwa tidak semua orang yang menerima kuesioner akan benar-benar menyelesaikannya. Rata-rata, hanya 30 sampai 50 persen kuesioner kertas dan email yang dikembalikan. Tingkat respons untuk kuesioner berbasis web cenderung jauh lebih rendah (seringkali hanya 5 hingga 30 persen).

Mengelola Masalah di Sesi JAD

Saya telah menjalankan lebih dari seratus sesi JAD dan telah mempelajari beberapa "trik fasilitator" standar. Berikut adalah beberapa masalah umum dan beberapa cara untuk mengatasinya.

- **Dominasi**. Fasilitator harus memastikan bahwa tidak ada orang yang mendominasi diskusi kelompok. Satu-satunya cara untuk menghadapi seseorang yang mendominasi adalah secara langsung. Selama istirahat, dekati orang tersebut, ucapkan terima kasih atas komentarnya yang berwawasan luas, dan mintalah orang tersebut untuk membantu Anda memastikan bahwa orang lain juga berpartisipasi.
- **Nonkontributor**. Menarik orang-orang yang telah berpartisipasi sangat sedikit adalah tantangan karena Anda ingin membawa mereka ke dalam percakapan sehingga mereka akan berkontribusi lagi. Pendekatan terbaik adalah mengajukan pertanyaan faktual langsung yang Anda yakin dapat mereka jawab. Dan itu membantu untuk mengajukan pertanyaan dalam cara yang panjang untuk memberi mereka waktu untuk berpikir. Misalnya, "Pat, saya tahu Anda sudah lama mengerjakan pengiriman pesanan. Anda mungkin sudah berada di departemen pengiriman lebih lama dari orang lain. Bisakah Anda membantu kami memahami dengan tepat apa yang terjadi ketika pesanan diterima dalam pengiriman?"
- **Diskusi sampingan**. Terkadang peserta terlibat dalam percakapan sampingan dan gagal memperhatikan kelompok. Solusi termudah adalah dengan berjalan dekat dengan orang-orang dan terus memfasilitasi tepat di depan mereka. Hanya sedikit orang yang akan melanjutkan konversi sampingan ketika Anda berada dua kaki dari mereka dan seluruh perhatian kelompok tertuju pada Anda dan mereka.
- **Agenda komidi putar**. Komedi-putar terjadi ketika seorang anggota grup terus kembali ke masalah yang sama setiap beberapa menit dan tidak mau melepaskannya. Salah satu solusinya adalah membiarkan orang tersebut memiliki waktu lima menit untuk mengoceh tentang masalah tersebut sementara Anda dengan hati-hati menuliskan setiap poin pada bagan flip atau file komputer. Flip chart atau file ini kemudian ditempelkan secara mencolok di dinding. Ketika orang tersebut mengemukakan masalah itu lagi, Anda menyela mereka, berjalan ke koran dan bertanya apa yang harus ditambahkan. Jika mereka menyebutkan sesuatu yang sudah ada dalam daftar, Anda segera menyela, menunjukkan bahwa itu ada di sana, dan menanyakan informasi lain apa yang perlu ditambahkan. Jangan biarkan mereka mengulangi poin yang sama, tetapi tulislah informasi baru.
- **Perjanjian kekerasan**. Beberapa ketidaksepakatan terburuk terjadi ketika peserta benar-benar setuju pada masalah tetapi tidak menyadari bahwa mereka setuju karena mereka menggunakan istilah yang berbeda. Contohnya adalah berdebat apakah gelas itu setengah kosong atau setengah penuh; mereka setuju pada fakta tetapi tidak dapat menyetujui kata-kata. Dalam hal ini, fasilitator harus menerjemahkan istilah

tersebut ke dalam kata-kata yang berbeda dan menemukan titik temu sehingga para pihak mengakui bahwa mereka benar-benar setuju.

- **Konflik yang belum terselesaikan**. Dalam beberapa kasus, peserta tidak setuju dan tidak dapat memahami bagaimana menentukan alternatif apa yang lebih baik. Anda dapat membantu dengan menyusun masalah. Tanyakan kriteria di mana kelompok akan mengidentifikasi alternatif yang baik (misalnya, "Misalkan ide ini benar-benar meningkatkan layanan pelanggan. Bagaimana saya mengenali layanan pelanggan yang ditingkatkan?"). Kemudian setelah Anda memiliki daftar kriteria, mintalah kelompok untuk menilai alternatif dengan menggunakan kriteria tersebut.
- **Konflik sejati**. Terkadang, terlepas dari setiap upaya, peserta tidak dapat menyetujui suatu masalah. Solusinya adalah menunda diskusi dan melanjutkan. Dokumentasikan isu tersebut sebagai isu terbuka dan daftarkan dengan jelas pada flip chart. Mintalah kelompok kembali ke masalah beberapa jam kemudian. Seringkali masalah akan teratasi dengan sendirinya saat itu dan Anda tidak membuang waktu untuk itu. Jika masalah tidak dapat diselesaikan nanti, pindahkan ke daftar masalah yang akan diputuskan oleh sponsor proyek atau anggota manajemen senior lainnya.
- **humor**. Humor adalah salah satu alat paling ampuh yang dimiliki seorang fasilitator dan karenanya harus digunakan dengan bijaksana. Humor JAD terbaik selalu dalam konteks; jangan pernah menceritakan lelucon tetapi ambil kesempatan untuk menemukan humor dalam situasi tersebut.

Karena informasi pada kuesioner tidak dapat segera diklarifikasi untuk responden yang bingung, mengembangkan pertanyaan yang baik sangat penting untuk kuesioner. Pertanyaan pada kuesioner harus ditulis dengan sangat jelas dan menyisakan sedikit ruang untuk kesalahpahaman, sehingga pertanyaan tertutup cenderung paling sering digunakan. Pertanyaan harus dengan jelas memungkinkan analis untuk memisahkan fakta dari opini. Pertanyaan opini sering menanyakan kepada responden sejauh mana mereka setuju atau tidak setuju (misalnya, Apakah masalah jaringan umum terjadi?), sedangkan pertanyaan faktual mencari nilai yang lebih tepat (misalnya, Seberapa sering masalah jaringan terjadi: satu jam sekali, sekali sehari, sekali seminggu?). Lihat Gambar 3-7 untuk pedoman desain kuesioner.

Mungkin masalah yang paling jelas—tetapi terkadang diabaikan—adalah memiliki pemahaman yang jelas tentang bagaimana informasi yang dikumpulkan dari kuesioner akan dianalisis dan digunakan. Masalah ini harus diatasi sebelum kuesioner dibagikan, karena sudah terlambat sesudahnya.

Pertanyaan harus relatif konsisten dalam gaya, sehingga responden tidak harus membaca instruksi untuk setiap pertanyaan sebelum menjawabnya. Biasanya merupakan praktik yang baik untuk mengelompokkan pertanyaan terkait bersama-sama agar lebih mudah dijawab. Beberapa ahli menyarankan bahwa kuesioner harus dimulai dengan pertanyaan-pertanyaan penting bagi responden, sehingga kuesioner segera menarik minat mereka dan mendorong mereka untuk menjawabnya. Mungkin langkah yang paling penting adalah meminta beberapa rekan untuk meninjau kuesioner dan kemudian mengujinya terlebih dahulu dengan beberapa orang yang diambil dari kelompok yang akan dikirimi kuesioner. Mengejutkan betapa seringnya pertanyaan yang tampaknya sederhana dapat disalahpahami.

Isu utama dalam mengelola kuesioner adalah membuat peserta mengisi kuesioner dan mengirimkannya kembali. Lusinan buku riset pemasaran telah ditulis tentang cara meningkatkan tingkat respons. Teknik yang umum digunakan termasuk menjelaskan dengan jelas mengapa kuesioner dilakukan dan mengapa responden dipilih, menyatakan tanggal

pengembalian kuesioner, menawarkan bujukan untuk melengkapi kuesioner (misalnya, pena gratis), dan menawarkan untuk memberikan ringkasan tanggapan kuesioner. Analis sistem memiliki teknik tambahan untuk meningkatkan tingkat respons di dalam organisasi, seperti secara pribadi membagikan kuesioner dan secara pribadi menghubungi mereka yang belum mengembalikannya setelah satu atau dua minggu, serta meminta supervisor responden untuk mengelola kuesioner dalam kelompok. pertemuan.

Membantu untuk memproses kuesioner yang dikembalikan dan mengembangkan laporan kuesioner segera setelah batas waktu kuesioner akan sangat membantu. Ini memastikan bahwa proses analisis berlangsung tepat waktu dan responden yang meminta salinan hasil menerimanya dengan segera.

- Mulailah dengan pertanyaan yang tidak mengancam dan menarik.
- Kelompokkan item ke dalam bagian yang koheren secara logis.
- Jangan meletakkan item penting di bagian paling akhir kuesioner.
- Jangan memadati halaman dengan terlalu banyak item.
- Hindari singkatan.
- Hindari item atau istilah yang bias atau sugestif.
- Beri nomor pada tiap pertanyaan untuk menghindari kebingungan.
- Pretest kuesioner untuk mengidentifikasi pertanyaan yang membingungkan.
- Berikan anonimitas kepada responden.

Gambar 3-7 Desain Kuesioner yang Baik

Analisis Dokumen

Tim proyek sering menggunakan analisis dokumen untuk memahami sistem apa adanya. Dalam keadaan ideal, tim proyek yang mengembangkan sistem yang ada akan menghasilkan dokumentasi yang kemudian diperbarui oleh semua proyek berikutnya. Dalam hal ini, tim proyek dapat memulai dengan meninjau dokumentasi dan memeriksa sistem itu sendiri.

Sayangnya, banyak sistem tidak didokumentasikan dengan baik karena tim proyek gagal mendokumentasikan proyek mereka di sepanjang jalan, dan ketika proyek selesai, tidak ada waktu untuk kembali dan mendokumentasikannya. Oleh karena itu, mungkin tidak ada banyak dokumentasi teknis tentang sistem saat ini yang tersedia, atau mungkin tidak berisi informasi terbaru tentang perubahan sistem terbaru. Namun, banyak dokumen bermanfaat memang ada dalam sebuah organisasi: laporan kertas, memorandum, manual kebijakan, manual pelatihan pengguna, bagan organisasi, formulir, dan, tentu saja, antarmuka pengguna dengan sistem yang ada.

Tetapi dokumen-dokumen ini hanya menceritakan sebagian dari cerita. Mereka mewakili sistem formal yang digunakan organisasi. Cukup sering, sistem yang nyata, atau informal, berbeda dari yang formal, dan perbedaan ini, terutama yang besar, memberikan indikasi kuat tentang apa yang perlu diubah. Misalnya, formulir atau laporan yang tidak pernah digunakan mungkin harus dihilangkan. Demikian pula, kotak atau pertanyaan pada formulir yang tidak pernah diisi (atau digunakan untuk tujuan lain) harus dipikirkan kembali. Lihat Gambar 3-8 untuk contoh bagaimana sebuah dokumen dapat diinterpretasikan.

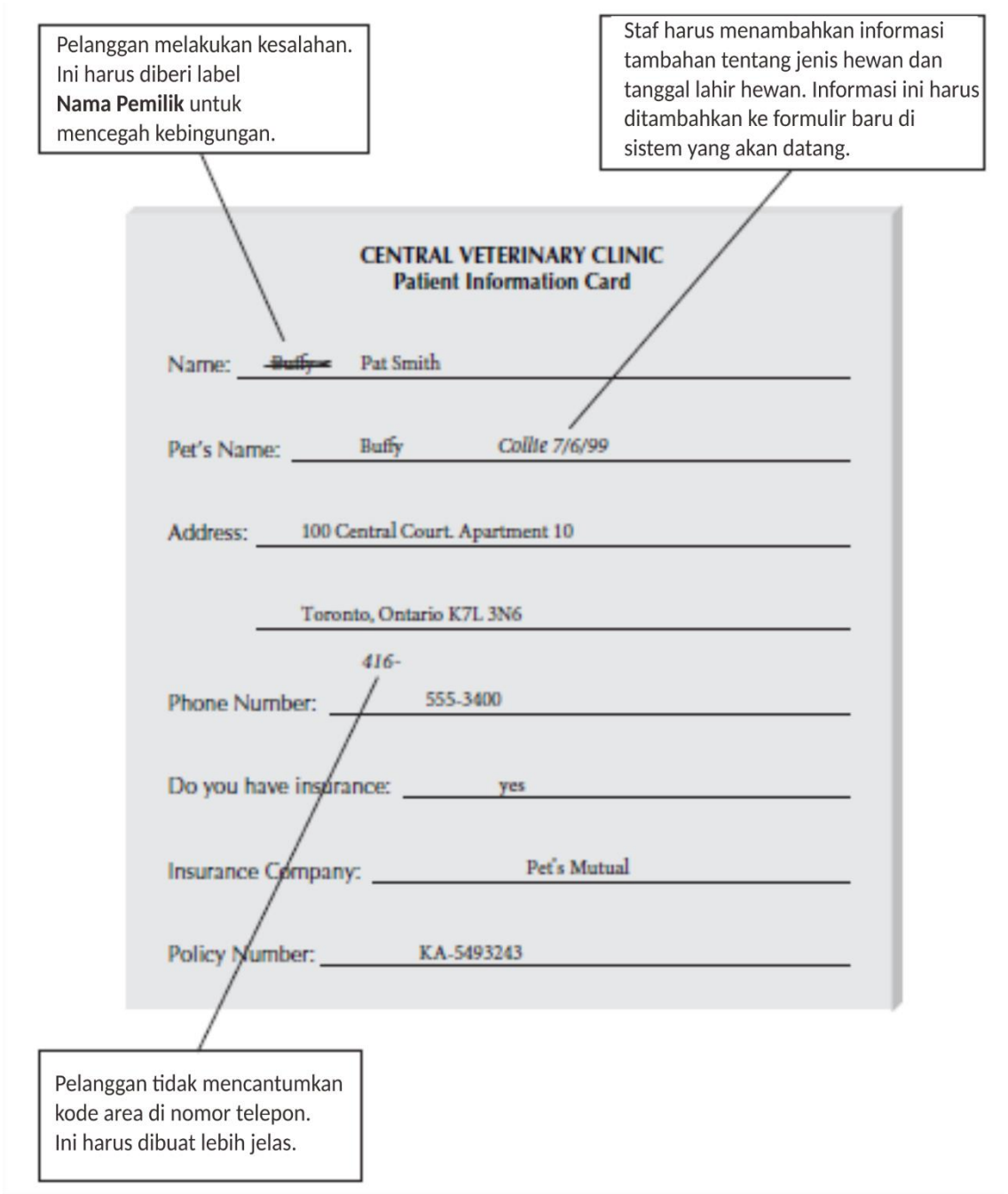
Indikasi paling kuat bahwa sistem perlu diubah adalah ketika pengguna membuat formulir mereka sendiri atau menambahkan informasi tambahan ke formulir yang sudah ada. Perubahan tersebut jelas menunjukkan perlunya perbaikan sistem yang ada. Dengan demikian, berguna untuk meninjau formulir kosong dan formulir yang telah diisi untuk mengidentifikasi

penyimpangan ini. Demikian juga, ketika pengguna mengakses beberapa laporan untuk memenuhi kebutuhan informasi mereka, itu adalah tanda yang jelas bahwa informasi baru atau format informasi baru diperlukan.

Observasi

Pengamatan, tindakan mengamati proses yang sedang dilakukan, adalah alat yang ampuh untuk mengumpulkan informasi tentang sistem saat in karena memungkinkan analis untuk melihat realitas situasi, daripada mendengarkan orang lain menggambarkannya dalam wawancara atau sesi JAD. Beberapa studi penelitian telah menunjukkan bahwa banyak manajer benar-benar tidak ingat bagaimana mereka bekerja dan bagaimana mereka mengalokasikan waktu mereka. (Cepat, berapa jam yang Anda habiskan minggu lalu untuk setiap kursus Anda?) Observasi adalah cara yang baik untuk memeriksa validitas informasi yang dikumpulkan dari sumber tidak langsung seperti wawancara dan kuesioner.

Dalam banyak hal, analis menjadi seorang antropolog saat dia berjalan melalui organisasi dan mengamati sistem bisnis sebagaimana fungsinya. Tujuannya adalah untuk tetap low profile, untuk tidak mengganggu mereka yang bekerja, dan untuk tidak mempengaruhi mereka yang sedang diamati. Meskipun demikian, penting untuk dipahami bahwa apa yang diamati oleh analis mungkin bukan rutinitas sehari-hari yang normal karena orang cenderung sangat berhati-hati dalam perilaku mereka ketika mereka sedang diawasi. Meskipun praktik normal mungkin melanggar aturan organisasi formal, pengamat tidak mungkin melihat ini. (Ingat bagaimana Anda mengemudi terakhir kali dan mobil polisi mengikuti Anda dari belakang?) Jadi, apa yang Anda lihat mungkin bukan yang Anda dapatkan.



Gambar 3-8 Melakukan Analisis Dokumen

Observasi sering digunakan untuk melengkapi informasi wawancara. Lokasi kantor seseorang dan perabotannya memberikan petunjuk tentang kekuatan dan pengaruh orang tersebut dalam organisasi dan dapat digunakan untuk mendukung atau menyangkal informasi yang diberikan dalam sebuah wawancara. Misalnya, seorang analis mungkin menjadi skeptis terhadap seseorang yang mengaku menggunakan sistem komputer yang ada secara ekstensif jika komputer tidak pernah dihidupkan saat analis berkunjung. Dalam kebanyakan kasus, observasi mendukung informasi yang diberikan pengguna dalam wawancara. Jika tidak, ini merupakan sinyal penting bahwa perhatian ekstra harus diberikan dalam menganalisis sistem bisnis.

Memilih Teknik yang Tepat

Masing-masing teknik pengumpulan kebutuhan yang dibahas sebelumnya memiliki kekuatan dan kelemahan. Tidak ada satu teknik yang selalu lebih baik dari yang lain, dan

dalam praktiknya sebagian besar proyek menggunakan kombinasi teknik. Oleh karena itu, penting untuk memahami kekuatan dan kelemahan masing-masing teknik dan kapan menggunakannya (lihat Gambar 3-9). Satu masalah yang tidak dibahas adalah pengalaman para analis. Secara umum, analisis dan observasi dokumen membutuhkan pelatihan paling sedikit, sedangkan sesi JAD adalah yang paling menantang.

	Interviews	Joint Application Design	Questionnaires	Document Analysis	Observation
Type of information	As-is, improvements, to-be	As-is, improvements, to-be	As-is, improvements	As-is	As-is
Depth of information	High	High	Medium	Low	Low
Breadth of information	Low	Medium	High	High	Low
Integration of information	Low	High	Low	Low	Low
User involvement	Medium	High	Low	Low	Low
Cost	Medium	Low to Medium	Low	Low	Low to Medium

Gambar 3-9 Tabel Persyaratan-Teknik Pengumpulan

Jenis Informasi. Karakteristik pertama adalah jenis informasi. Beberapa teknik lebih cocok untuk digunakan pada berbagai tahap proses analisis, apakah memahami sistem apa adanya, mengidentifikasi perbaikan, atau mengembangkan sistem yang akan datang. Wawancara dan JAD biasanya digunakan dalam ketiga tahap tersebut. Sebaliknya, analisis dan observasi dokumen biasanya paling membantu untuk memahami apa adanya, meskipun terkadang mereka memberikan informasi tentang masalah saat ini yang perlu diperbaiki. Kuesioner sering digunakan untuk mengumpulkan informasi tentang sistem saat in serta informasi umum tentang perbaikan.

Kedalaman Informasi. Kedalaman informasi mengacu pada seberapa kaya dan detail informasi yang biasanya dihasilkan oleh teknik dan sejauh mana teknik itu berguna untuk memperoleh tidak hanya fakta dan opini tetapi juga pemahaman tentang mengapa fakta dan opini itu ada. Wawancara dan sesi JAD sangat berguna untuk memberikan kedalaman informasi yang kaya dan terperinci dan membantu analis untuk memahami alasan di baliknya. Di sisi lain, analisis dan observasi dokumen berguna untuk memperoleh fakta, tetapi sedikit di luar itu. Kuesioner dapat memberikan kedalaman informasi yang sedang, mengumpulkan fakta dan opini dengan sedikit pemahaman tentang mengapa mereka ada.

Luasnya Informasi. Keluasan informasi mengacu pada jangkauan informasi dan sumber informasi yang dapat dengan mudah dikumpulkan dengan menggunakan teknik yang dipilih. Kuesioner dan analisis dokumen keduanya dengan mudah mampu mengumpulkan berbagai informasi dari sejumlah besar sumber informasi. Sebaliknya, wawancara dan observasi mengharuskan analis untuk mengunjungi setiap sumber informasi secara individual dan, oleh karena itu, membutuhkan lebih banyak waktu. Sesi JAD berada di tengah-tengah karena banyak sumber informasi disatukan pada saat yang bersamaan.

Integrasi Informasi. Salah satu aspek yang paling menantang dari pengumpulan persyaratan adalah mengintegrasikan informasi dari sumber yang berbeda. Sederhananya, orang yang berbeda dapat memberikan informasi yang saling bertentangan. Menggabungkan informasi ini dan mencoba menyelesaikan perbedaan pendapat atau fakta biasanya sangat memakan waktu karena itu berarti menghubungi setiap sumber informasi secara bergantian, menjelaskan perbedaan, dan berusaha memperbaiki informasi. Dalam banyak kasus, individu salah memahami bahwa analis menantang informasinya, padahal sebenarnya pengguna lain dalam organisasi yang melakukannya. Hal ini dapat membuat pengguna menjadi defensif dan mempersulit penyelesaian perbedaan.

Semua teknik mengalami masalah integrasi sampai tingkat tertentu, tetapi sesi JAD dirancang untuk meningkatkan integrasi karena semua informasi terintegrasi saat dikumpulkan, bukan setelahnya. Jika dua pengguna memberikan informasi yang bertentangan, konflik menjadi jelas, seperti halnya sumber konflik. Integrasi langsung informasi adalah satu-satunya manfaat terpenting JAD yang membedakannya dari teknik lain, dan inilah mengapa sebagian besar organisasi menggunakan JAD untuk proyek-proyek penting.

Keterlibatan Pengguna. Keterlibatan pengguna mengacu pada jumlah waktu dan energi yang harus dicurahkan oleh pengguna yang dituju dari sistem baru untuk proses analisis. Secara umum disepakati bahwa ketika pengguna menjadi lebih terlibat dalam proses analisis, peluang keberhasilan meningkat. Namun, keterlibatan pengguna dapat memiliki biaya yang signifikan, dan tidak semua pengguna bersedia menyumbangkan waktu dan energi yang berharga. Kuesioner, analisis dokumen, dan observasi menempatkan beban paling sedikit pada pengguna, sedangkan sesi JAD membutuhkan upaya terbesar.

Biaya. Biaya selalu menjadi pertimbangan penting. Secara umum, kuesioner, analisis dokumen, dan observasi merupakan teknik yang murah (walaupun observasi dapat memakan waktu yang lama). Biaya rendah tidak berarti bahwa mereka lebih atau kurang efektif daripada teknik lainnya. Wawancara dan sesi JAD umumnya memiliki biaya yang moderat. Secara umum, sesi JAD pada awalnya jauh lebih mahal, karena mengharuskan banyak pengguna untuk absen dari kantor mereka untuk jangka waktu yang signifikan, dan sering kali melibatkan konsultan yang dibayar tinggi. Namun, sesi JAD secara signifikan mengurangi waktu yang dihabiskan dalam integrasi informasi dan dengan demikian dapat mengurangi biaya dalam jangka panjang.

Menggabungkan Teknik. Dalam prakteknya, pengumpulan kebutuhan menggabungkan serangkaian teknik yang berbeda. Sebagian besar analis memulai dengan menggunakan wawancara dengan manajer senior untuk mendapatkan pemahaman tentang proyek dan masalah gambaran besarnya. Dari wawancara ini, menjadi jelas apakah perubahan besar atau kecil diantisipasi. Wawancara ini sering diikuti dengan analisis dokumen dan kebijakan untuk mendapatkan pemahaman tentang sistem apa adanya. Biasanya wawancara datang berikutnya untuk mengumpulkan sisa informasi yang diperlukan untuk gambaran apa adanya.

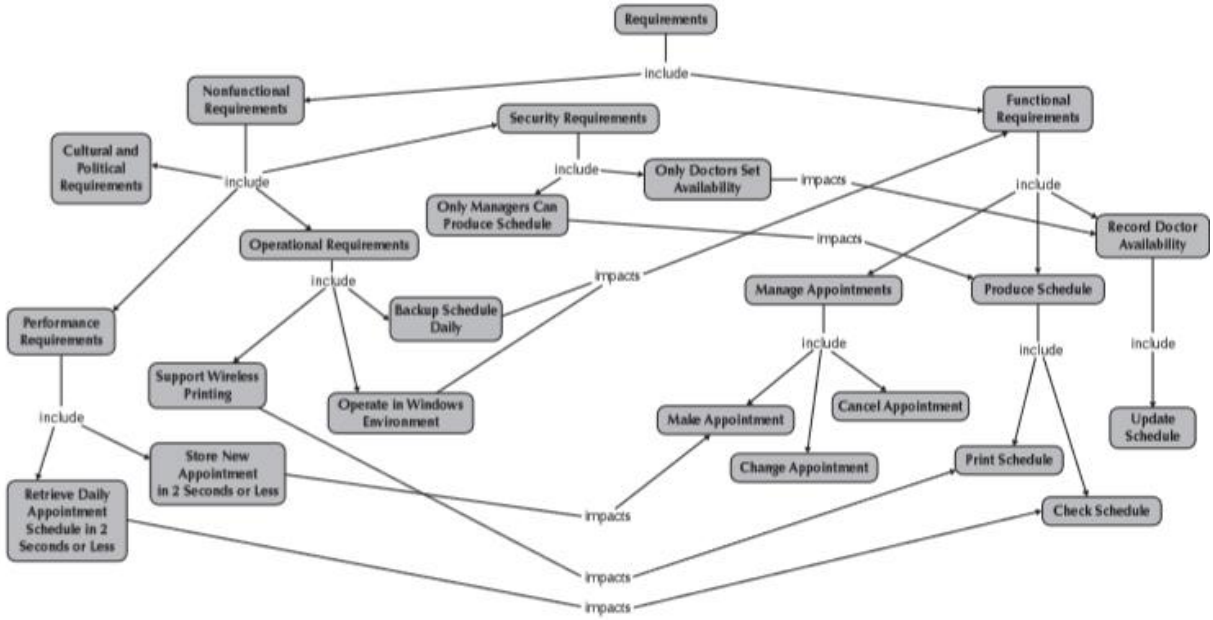
Dalam pengalaman kami, mengidentifikasi peningkatan paling sering dilakukan menggunakan sesi JAD karena sesi JAD memungkinkan pengguna dan pemegang kepentingan utama untuk bekerja sama melalui teknik analisis dan mencapai pemahaman bersama tentang kemungkinan sistem yang akan datang. Kadang-kadang, sesi JAD ini diikuti dengan kuesioner yang dikirim ke pengguna atau calon pengguna yang lebih luas untuk melihat apakah pendapat mereka yang berpartisipasi dalam sesi JAD dibagikan secara luas.

Mengembangkan konsep untuk sistem yang akan datang sering dilakukan melalui wawancara dengan manajer senior, diikuti dengan sesi JAD dengan pengguna dari semua tingkatan untuk memastikan bahwa kebutuhan utama dari sistem baru dipahami dengan baik.

3.6 TEKNIK DOKUMENTASI PERSYARATAN ALTERNATIF

Beberapa teknik pengumpulan dan dokumentasi persyaratan yang sangat berguna lainnya termasuk pembuatan prototipe sekali pakai, Use-Case, kartu CRC permainan peran dengan skenario berbasis Use-Case, pemetaan konsep, dan perekaman cerita pengguna pada kartu cerita dan daftar tugas. Prototyping sekali pakai dijelaskan di Bab 1. Intinya, prototipe sekali pakai dibuat untuk lebih memahami beberapa aspek dari sistem baru. Dalam banyak kasus, mereka digunakan untuk menguji beberapa aspek teknis dari persyaratan nonfungsional, seperti menghubungkan workstation klien ke server. Jika Anda belum pernah melakukan ini

sebelumnya, akan jauh lebih mudah untuk mengembangkan sistem contoh yang sangat kecil untuk menguji desain koneksi yang diperlukan dari workstation klien ke server daripada mencoba melakukannya pertama kali dengan penuh -sistem yang ditiup. Throwaway prototyping sangat berguna dalam mendesain antarmuka pengguna (lihat Bab 10).



Gambar 3-10 Contoh Peta Konsep Persyaratan

Use-Case, seperti yang dijelaskan dalam Bab 1, adalah pendekatan mendasar yang digunakan Proses Terpadu dan Bahasa Pemodelan Terpadu (UML) untuk mendokumentasikan dan mengumpulkan persyaratan fungsional. Kami menjelaskannya di Bab 4. Kartu CRC bermain peran dengan skenario berbasis Use-Case sangat berguna saat membuat model fungsional (lihat Bab 4), struktural (lihat Bab 5), dan perilaku (lihat Bab 6). Kami menjelaskan pendekatan ini dalam Bab 5. Sisa dari bagian ini menjelaskan penggunaan pemetaan konsep yang merekam cerita pengguna pada kartu cerita dan daftar tugas.

Peta konsep

Peta konsep mewakili hubungan yang bermakna antar konsep. Mereka berguna untuk memfokuskan individu pada sejumlah kecil ide-ide utama di mana mereka harus berkonsentrasi. Peta konsep pada dasarnya adalah representasi node-and-arc, di mana node mewakili persyaratan individu dan busur mewakili hubungan di antara persyaratan. Setiap busur diberi label dengan nama hubungan. Peta konsep juga telah direkomendasikan sebagai teknik yang memungkinkan untuk mendukung persyaratan pemodelan untuk pengembangan sistem berorientasi objek dan sistem manajemen pengetahuan. Pemetaan konsep adalah teknik psikologi pendidikan yang telah digunakan di sekolah, perusahaan, dan lembaga perawatan kesehatan untuk memfasilitasi pembelajaran, pemahaman, dan penciptaan pengetahuan. Keuntungan dari pendekatan pemetaan konsep untuk mewakili persyaratan atas pendekatan tekstual yang khas (lihat Gambar 3-1) adalah bahwa peta konsep tidak terbatas pada representasi hierarkis. Peta konsep memungkinkan hubungan antara persyaratan fungsional dan nonfungsional untuk diwakili secara eksplisit.

Gambar 3-10 menunjukkan peta konsep yang menggambarkan informasi yang terkandung dalam definisi persyaratan yang ditunjukkan pada Gambar 3-1. Dengan menggunakan peta konsep untuk mewakili persyaratan alih-alih pendekatan tekstual, hubungan antara persyaratan fungsional dan nonfungsional dapat dibuat eksplisit. Misalnya, dua persyaratan keamanan Ketersediaan Hanya Dokter yang Ditetapkan dan Hanya Manajer

yang Dapat Memproduksi Jadwal secara eksplisit terkait dengan persyaratan fungsional Ketersediaan Dokter Rekam dan Jadwal Produksi. Ini sangat sulit untuk direpresentasikan dalam versi teks saja dari definisi persyaratan. Selain itu, dengan membuat pengguna dan analis fokus pada tata letak grafis peta, persyaratan tambahan dapat ditemukan. Salah satu masalah yang jelas dengan pendekatan ini adalah bahwa jika jumlah persyaratan menjadi banyak dan hubungan di antara mereka menjadi kompleks, maka jumlah node dan busur akan menjadi begitu terjalin sehingga keuntungan untuk dapat melihat hubungan secara eksplisit akan hilang. Namun, dengan menggabungkan representasi teks dan peta konsep, dimungkinkan untuk memanfaatkan kekuatan representasi tekstual dan grafis untuk lebih sepenuhnya mewakili persyaratan.

Cerita Pengguna

Cerita pengguna, bersama dengan kartu cerita dan daftar tugas terkait, dikaitkan dengan pendekatan pengembangan tangkas. Cerita pengguna telah terbukti sangat berguna dalam mengumpulkan persyaratan dengan cara yang tidak mengancam yang menghormati sudut pandang pengguna. Mereka biasanya ditangkap menggunakan kartu cerita (kartu indeks) dan dicatat pada daftar tugas (atau dari perspektif Scrum, pada backlog produk). Baik kartu cerita maupun daftar tugas dianggap sebagai pendekatan yang ringan untuk mendokumentasikan dan mengumpulkan persyaratan. Cerita mencakup persyaratan fungsional dan nonfungsional. Misalnya, sehubungan dengan contoh janji temu dokter, cerita berbasis kebutuhan fungsional dapat berupa:

> Sebagai sekretaris, saya ingin dapat menjadwalkan janji temu untuk pasien kami sehingga kami dapat memenuhi kebutuhan pasien kami.

Sementara cerita berbasis kebutuhan nonfungsional operasional dapat berupa:

> Sebagai seorang sekretaris, saya ingin dapat mencetak jadwal harian menggunakan teknologi nirkabel sehingga semua pencetakan dapat dilakukan menggunakan printer bersama tanpa harus berurusan dengan kabel printer yang menghubungkan semua komputer ke printer.

Setelah cerita ditulis, dibahas untuk menentukan jumlah upaya yang diperlukan untuk mengimplementasikannya. Selama diskusi, daftar tugas dibuat untuk cerita. Jika cerita dianggap terlalu besar—misalnya, ada terlalu banyak tugas di daftar tugas—cerita dibagi menjadi beberapa cerita yang masing-masing direkam pada kartu cerita sendiri dan tugas dialokasikan di seluruh cerita baru. Setelah serangkaian tugas diidentifikasi dengan cerita, cerita dan tugasnya ditempel di dinding bersama sehingga semua anggota tim pengembangan dapat melihat persyaratannya. Cerita dapat diprioritaskan berdasarkan kepentingan dengan menempatkan peringkat pada kartu. Cerita juga dapat dievaluasi untuk tingkat risiko yang terkait dengannya. Tingkat kepentingan dan jumlah risiko yang terkait dengan cerita dapat digunakan untuk membantu memilih persyaratan mana yang akan diterapkan terlebih dahulu. Keuntungan menggunakan kartu cerita dan daftar tugas untuk mendokumentasikan persyaratan adalah teknologinya sangat rendah, sentuhan tinggi, mudah diperbarui, dan sangat portabel.

3.7 PROPOSAL SISTEM

Proposal sistem menyatukan ke dalam satu dokumen komprehensif materi yang dibuat selama perencanaan dan analisis. Proposal sistem biasanya mencakup ringkasan eksekutif, permintaan sistem, rencana kerja, analisis kelayakan, definisi persyaratan, dan model yang berkembang yang menggambarkan sistem baru. Model yang berkembang meliputi model fungsional (lihat Bab 4), model struktural (lihat Bab 5), dan model perilaku (lihat Bab 6).

Ringkasan eksekutif memberikan semua informasi penting dalam bentuk yang sangat ringkas. Ini dapat dianggap sebagai ringkasan dari proposal lengkap. Tujuannya adalah untuk memungkinkan seorang eksekutif yang sibuk membaca dengan cepat dan menentukan bagian mana dari proposal yang harus dia lalui dengan lebih teliti. Ringkasan eksekutif biasanya tidak lebih dari satu halaman. Gambar 3-11 menyediakan template untuk proposal sistem dan referensi di mana bagian lain dari proposal dijelaskan.

1. **Daftar isi**
2. **Ringkasan Eksekutif**

   Ringkasan semua informasi penting dalam proposal sehingga eksekutif yang sibuk dapat membacanya dengan cepat dan memutuskan bagian mana dari proposal yang akan dibaca secara lebih mendalam.

3. **Permintaan Sistem**

   Formulir permintaan sistem yang direvisi (lihat Bab 2).

4. **Rencana kerja**

   Rencana kerja asli, direvisi setelah analisis selesai (lihat Bab 2).

5. **Analisis Kelayakan**

   Analisis kelayakan yang direvisi, dengan menggunakan informasi dari analisis (lihat Bab 2).

6. **Definisi Persyaratan**

   Daftar kebutuhan bisnis fungsional dan nonfungsional untuk sistem (bab ini).

7. **Model Fungsional**

   Diagram aktivitas, satu set deskripsi Use-Case, dan diagram Use-Case yang menggambarkan proses dasar atau fungsionalitas eksternal yang perlu didukung sistem (lihat Bab 4).

8. **Model Struktural**

   Satu set kartu CRC, diagram kelas, dan diagram objek yang menggambarkan aspek struktural dari sistem yang akan dibuat (lihat Bab 5). Ini juga dapat mencakup model struktural dari sistem saat in saat ini yang akan diganti.

9. **Model Perilaku**

   Satu set diagram urutan, diagram komunikasi, mesin status perilaku, dan matriks CRUDE yang menggambarkan perilaku internal sistem yang akan datang (lihat Bab 6). Ini mungkin termasuk model perilaku dari sistem saat in yang akan diganti.

10. **Lampiran**

    Ini berisi materi tambahan yang relevan dengan proposal, sering digunakan untuk mendukung sistem yang direkomendasikan. Ini mungkin termasuk hasil survei kuesioner atau wawancara, laporan dan statistik industri, dan sebagainya.

Gambar 3-11 Template Proposal Sistem

Pertanyaan

1. Apa kiriman utama yang dibuat selama analisis? Apa hasil akhir dari analisis, dan apa isinya?
2. Apa perbedaan antara sistem saat in dan sistem yang akan datang?
3. Apa tujuan dari definisi kebutuhan?
4. Apa tiga langkah dasar dari proses analisis? Langkah mana yang terkadang dilewati atau dilakukan secara sepintas? Mengapa?
5. Bandingkan dan kontraskan analisis masalah dan analisis akar masalah. Dalam kondisi apa Anda akan menggunakan analisis masalah? Dalam kondisi apa Anda akan menggunakan analisis akar penyebab?
6. Bandingkan dan kontraskan antara analisis durasi dan penetapan biaya berdasarkan aktivitas.
7. Jelaskan lima langkah utama dalam melakukan wawancara.
8. Jelaskan perbedaan antara pertanyaan tertutup, pertanyaan terbuka, dan pertanyaan penyelidikan. Kapan Anda akan menggunakan masing-masing?
9. Jelaskan perbedaan antara wawancara tidak terstruktur dan wawancara terstruktur. Kapan Anda akan menggunakan setiap pendekatan?
10. Jelaskan perbedaan antara pendekatan wawancara top-down dan bottom-up. Kapan Anda akan menggunakan setiap pendekatan?
11. Bagaimana peserta dipilih untuk wawancara dan sesi JAD?
12. Bagaimana cara membedakan antara fakta dan opini? Mengapa keduanya bisa bermanfaat?
13. Jelaskan lima langkah utama dalam melakukan sesi JAD.
14. Bagaimana fasilitator JAD berbeda dari juru tulis?
15. Apa tiga hal utama yang dilakukan fasilitator dalam melakukan sesi JAD?
16. Apa itu e-JAD, dan mengapa sebuah perusahaan tertarik untuk menggunakannya?
17. Bagaimana merancang pertanyaan untuk kuesioner berbeda dari merancang pertanyaan untuk wawancara atau sesi JAD?
18. Berapa tingkat respons khas untuk kuesioner, dan bagaimana Anda dapat meningkatkannya?
19. Apa itu analisis dokumen?
20. Apa perbedaan sistem formal dengan sistem informal? Bagaimana analisis dokumen membantu Anda memahami keduanya?
21. Apa aspek utama dari menggunakan observasi dalam proses pengumpulan informasi?
22. Jelaskan faktor-faktor yang dapat digunakan untuk memilih teknik pengumpulan informasi.
23. Apa keuntungan utama yang dimiliki peta konsep dibandingkan teknik dokumen persyaratan tekstual tradisional?
24. Apa saja keuntungan menggunakan kartu cerita dan daftar tugas sebagai teknik pengumpulan-persyaratan dan dokumentasi?
25. Informasi apa yang biasanya disertakan dalam proposal sistem?
26. Apa tujuan dari ringkasan eksekutif dari proposal sistem?

Latihan

A. Tinjau situs web Amazon.com. Kembangkan definisi persyaratan untuk situs. Buat daftar persyaratan bisnis fungsional yang dipenuhi sistem. Apa saja jenis kebutuhan bisnis nonfungsional yang dipenuhi sistem? Berikan contoh untuk setiap jenis.
B. Misalkan Anda akan membangun sistem baru yang mengotomatisasi atau meningkatkan proses wawancara untuk departemen layanan karir di kampus Anda. Mengembangkan definisi kebutuhan untuk sistem baru. Sertakan persyaratan sistem

fungsional dan nonfungsional. Berpura-puralah seakan Anda akan merilis sistem dalam tiga versi yang berbeda. Prioritaskan persyaratan yang sesuai.

C. Jelaskan dalam istilah yang sangat umum proses bisnis apa adanya untuk mendaftar kelas di universitas Anda. Berkolaborasilah dengan siswa lain di kelas Anda, dan evaluasi prosesnya menggunakan analisis masalah dan analisis akar masalah. Berdasarkan pekerjaan Anda, buatlah daftar beberapa perbaikan yang telah Anda identifikasi.
D. Jelaskan dalam istilah yang sangat umum proses bisnis apa adanya untuk melamar masuk di universitas Anda. Berkolaborasilah dengan mahasiswa lain di kelas Anda, dan evaluasi prosesnya menggunakan benchmarking informal. Berdasarkan pekerjaan Anda, buatlah daftar beberapa perbaikan yang telah Anda identifikasi.
E. Jelaskan dalam istilah yang sangat umum proses bisnis apa adanya untuk mendaftar kelas di universitas Anda. Berkolaborasi dengan siswa lain di kelas Anda, dan evaluasi prosesnya menggunakan penghapusan aktivitas. Berdasarkan pekerjaan Anda, buatlah daftar beberapa perbaikan yang telah Anda identifikasi.
F. Misalkan universitas Anda mengalami peningkatan dramatis dalam pendaftaran dan mengalami kesulitan menemukan kursi yang cukup dalam kursus untuk siswa. Lakukan analisis teknologi untuk mengidentifikasi cara baru untuk membantu siswa menyelesaikan studi dan lulus.
G. Misalkan Anda adalah analis yang ditugaskan untuk mengembangkan sistem baru untuk toko buku universitas sehingga siswa dapat memesan buku secara online dan mengirimkannya ke asrama atau perumahan di luar kampus. Teknik pengumpulan persyaratan apa yang akan Anda gunakan? Jelaskan secara rinci bagaimana Anda akan menerapkan teknik.
H. Misalkan Anda adalah analis yang ditugaskan untuk mengembangkan sistem baru untuk membantu manajer senior membuat keputusan strategis yang lebih baik. Teknik pengumpulan persyaratan apa yang akan Anda gunakan? Jelaskan secara rinci bagaimana Anda akan menerapkan teknik.
I. Temukan pasangan dan wawancarai satu sama lain tentang tugas apa yang dilakukan masing-masing dalam pekerjaan terakhir yang Anda pegang (penuh waktu, paruh waktu, dulu, atau sekarang). Jika Anda belum pernah bekerja sebelumnya, anggaplah pekerjaan Anda adalah menjadi mahasiswa. Sebelum Anda melakukan ini, kembangkan rencana wawancara singkat. Setelah pasangan Anda mewawancarai Anda, kenali jenis wawancara, pendekatan wawancara, dan jenis pertanyaan yang digunakan.
J. Temukan sekelompok mahasiswa dan jalankan sesi JAD selama enam puluh menit untuk meningkatkan hubungan alumni di universitas Anda. Kembangkan rencana JAD singkat, pilih dua teknik yang akan membantu mengidentifikasi perbaikan, dan kemudian mengembangkan agenda. Lakukan sesi menggunakan agenda, dan tulis laporan pasca-sesi Anda.
K. Temukan kuesioner di Web yang telah dibuat untuk menangkap informasi pelanggan. Jelaskan tujuan survei, cara pertanyaan disusun, dan bagaimana pertanyaan disusun. Bagaimana itu bisa ditingkatkan? Bagaimana tanggapan akan dianalisis?
L. Kembangkan kuesioner yang akan membantu mengumpulkan informasi mengenai proses di restoran populer atau kafetaria kampus (misalnya, pemesanan, layanan pelanggan). Berikan kuesioner kepada sepuluh hingga lima belas siswa, analisis tanggapan, dan tulis laporan singkat yang menjelaskan hasilnya.
M. Hubungi departemen layanan karir di universitas Anda, dan temukan semua dokumen terkait yang dirancang untuk membantu siswa menemukan pekerjaan tetap dan/atau paruh waktu. Menganalisis dokumen dan menulis laporan singkat.

Minicase

1. Asosiasi Pemadam Kebakaran memiliki keanggotaan 15.000. Tujuan dari organisasi ini adalah untuk memberikan beberapa dukungan keuangan kepada keluarga anggota pemadam kebakaran yang meninggal dan untuk menyelenggarakan konferensi setiap tahun yang menyatukan petugas pemadam kebakaran dari seluruh negara bagian. Anggota ditagih iuran dan panggilan setiap tahun. Panggilan adalah dana tambahan yang diperlukan untuk mengurus pembayaran yang dilakukan kepada keluarga anggota yang meninggal. Pekerjaan pembukuan untuk asosiasi tersebut ditangani oleh bendahara terpilih, Andre, meskipun diketahui secara luas bahwa istrinya, Cintya, yang melakukan semua pekerjaan itu.

   Andre berjalan tanpa lawan setiap tahun di pemilihan, karena tidak ada yang mau mengambil alih pekerjaan pelacakan keanggotaan yang membosankan dan memakan waktu. Andre digaji Rp.112.000.000 per tahun, tetapi istrinya menghabiskan lebih dari dua puluh jam per minggu untuk pekerjaan itu. Organisasi, bagaimanapun, tidak senang dengan kinerja mereka.

   Sistem komputer digunakan untuk melacak penagihan dan penerimaan dana. Sistem ini dikembangkan pada tahun 1984 oleh seorang mahasiswa ilmu komputer dan ayahnya. Sistem ini adalah sistem berbasis DOS yang ditulis menggunakan dBase 3. Masalah paling mendesak yang dihadapi bendahara dan istrinya adalah kenyataan bahwa paket perangkat lunak tidak ada lagi, dan tidak ada orang di sekitar yang tahu cara memelihara sistem. Satu kueri, khususnya, membutuhkan waktu tujuh belas jam untuk dijalankan. Selama bertahun-tahun, mereka menghindari menjalankan kueri ini, meskipun informasi di dalamnya akan sangat berguna. Pertanyaan dari anggota mengenai pernyataan mereka tidak dapat dengan mudah dijawab. Biasanya Andre atau Cintya hanya mencatat pertanyaan dan membalas telepon dengan jawabannya. Terkadang dibutuhkan tiga sampai lima jam untuk menemukan informasi yang dibutuhkan untuk menjawab pertanyaan tersebut. Seringkali, mereka harus melakukan perhitungan secara manual karena sistem tidak diprogram untuk menangani jenis kueri tertentu. Ketika informasi anggota dimasukkan ke dalam sistem, setiap bidang disajikan satu per satu, yang membuat sangat sulit untuk kembali ke bidang dan memperbaiki nilai yang dimasukkan. Terkadang ada anggota baru yang masuk tetapi menghilang dari catatan. Laporan keanggotaan yang digunakan dalam materi konferensi tidak mengurutkan anggota berdasarkan kota. Hanya kota yang terdaftar dalam urutan yang benar.

   Strategi analisis kebutuhan apa yang akan Anda rekomendasikan untuk situasi ini? Jelaskan jawabanmu.

2. Rian, manajer proyek IS, hampir siap untuk berangkat ke pertemuan mendesak yang dipanggil oleh Devian, manajer operasi manufaktur. Sebuah proyek besar yang disponsori oleh Devian baru-baru ini menyelesaikan rintangan persetujuan, dan Rian membantu membawa proyek tersebut melalui inisiasi proyek. Sekarang setelah komite persetujuan telah memberikan lampu hijau, Brian telah mengerjakan rencana analisis proyek.

   Suatu malam, saat bermain golf dengan seorang teman yang bekerja di departemen operasi manufaktur, Rian mengetahui bahwa Devian ingin mendorong kerangka waktu proyek dari perkiraan awal Rian selama tiga belas bulan. Teman Rian mendengar Devian berkata, “Saya tidak mengerti mengapa tim proyek IS itu perlu menghabiskan waktu untuk menganalisis berbagai hal. Mereka punya dua minggu yang dijadwalkan hanya untuk melihat sistem yang ada! Itu sepertinya benar-benar sia-sia. Saya ingin tim itu terus membangun sistem saya.”

Karena Rian memiliki sedikit pengetahuan orang dalam tentang agenda Devian untuk pertemuan ini, dia telah mempertimbangkan bagaimana menangani Devian. Apa yang Anda sarankan Rian memberitahu Devian?

3. Dani baru-baru ini ditugaskan ke tim proyek yang akan mengembangkan sistem manajemen toko ritel baru untuk rantai toko sandwich bawah laut. Dani memiliki beberapa tahun pengalaman dalam pemrograman, tetapi dia belum melakukan banyak analisis dalam karirnya. Dia sedikit gugup tentang pekerjaan baru yang akan dia lakukan, tetapi dia yakin dia bisa menangani tugas apa pun yang diberikan kepadanya.

   Salah satu tugas pertama Dani adalah mengunjungi salah satu toko roti di sebuah lokasi yang ditentukan. Dani berencana tiba di toko sekitar tengah hari, tetapi dia memilih toko di area kota yang tidak dia kenal, dan karena kemacetan lalu lintas dan kesulitan menemukan toko, dia tidak datang sampai 1:30. Manajer toko tidak mengharapkan dia dan menolak untuk membiarkan orang asing di belakang meja sampai Dani menghubungi sponsor proyek (direktur manajemen toko) di kantor pusat perusahaan untuk memverifikasi siapa dia dan apa tujuannya.

   Setelah akhirnya mendapatkan izin untuk mengamati, Dani menempatkan dirinya secara terbuka di area kerja di belakang konter sehingga dia bisa melihat semuanya. Staf harus bermanuver di sekelilingnya saat mereka melakukan tugas mereka, tetapi hanya ada tabrakan kecil sesekali. Dani memperhatikan bahwa staf toko tampaknya melakukan pekerjaan mereka dengan sangat lambat dan sengaja, tetapi dia menduga itu karena tokonya tidak terlalu sibuk. Pada awalnya, Dani menanyai setiap pekerja tentang apa yang dia lakukan, tetapi manajer toko akhirnya memintanya untuk tidak terlalu mengganggu pekerjaan mereka—dia mengganggu layanan mereka kepada pelanggan.

   Pada pukul 3:30, Dani sedikit bosan. Dia memutuskan untuk pergi, berpikir dia bisa kembali ke kantor dan menyiapkan laporannya sebelum pukul 5:00 hari itu. Dia yakin pemimpin timnya akan senang dengan penyelesaian tugasnya yang cepat. Saat mengemudi, dia merenungkan, "Tidak banyak yang bisa dikatakan dalam laporan ini. Yang mereka lakukan hanyalah menerima pesanan, membuat cake, mengumpulkan pembayaran, dan menyerahkan pesanan. Ini sangat sederhana!" Keyakinan Dani dalam keterampilan analitisnya melonjak saat dia mengantisipasi pujian pemimpin timnya.

   Ketika kembali ke toko, manajer toko menggelengkan kepalanya, berkomentar kepada stafnya, "Dia datang ke sini pada waktu paling lambat pada hari yang paling lambat dalam seminggu. Dia bahkan tidak pernah melihat semua pekerjaan yang saya lakukan di ruang belakang saat dia di sini — meringkas penjualan kemarin, memeriksa inventaris yang ada, membuat pesanan pasokan untuk akhir pekan. . . ditambah dia bahkan tidak pernah mempertimbangkan prosedur pembukaan dan penutupan toko kami. Saya benci berpikir bahwa sistem manajemen toko baru akan dibangun oleh orang seperti itu. Sebaiknya saya menghubungi Anton [direktur manajemen toko] dan memberi tahu dia apa yang terjadi di sini hari ini."

   Evaluasilah perilaku Dani terhadap sebuah tugas.

4. Ani telah diberi tugas untuk melakukan survei terhadap pegawai penjualan yang akan menggunakan sistem entri pesanan baru yang sedang dikembangkan untuk perusahaan katalog produk rumah tangga. Tujuan dari survei ini adalah untuk mengidentifikasi pendapat panitera tentang kekuatan dan kelemahan sistem saat ini. Ada sekitar 50

pegawai yang bekerja di tiga kota yang berbeda, jadi survei sepertinya merupakan cara yang ideal untuk mengumpulkan informasi yang dibutuhkan dari pegawai.

Ani mengembangkan kuesioner dengan hati-hati dan mengujinya terlebih dahulu pada beberapa supervisor penjualan yang tersedia di kantor pusat perusahaan. Setelah merevisinya berdasarkan saran mereka, dia mengirim kuesioner versi kertas ke setiap petugas, meminta agar kuesioner itu dikembalikan dalam waktu satu minggu. Setelah satu minggu, dia hanya mengembalikan tiga kuesioner yang telah diisi. Setelah seminggu lagi, Ani hanya menerima dua kuesioner lagi. Merasa agak putus asa, Ani kemudian mengirimkan kuesioner versi e-mail, sekali lagi kepada semua pegawai, meminta mereka untuk menanggapi kuesioner melalui e-mail sesegera mungkin. Dia menerima dua e-mail kuesioner dan tiga pesan dari pegawai yang telah menyelesaikan versi kertas yang menyatakan jengkel karena diganggu dengan kuesioner yang sama untuk kedua kalinya. Pada titik ini, Ani hanya memiliki tingkat respons 14 persen, yang dia yakin tidak akan menyenangkan pemimpin timnya. Saran apa yang Anda miliki yang dapat meningkatkan tingkat respons Ani terhadap kuesioner?

BAB 4

PROSES BISNIS DAN PERMODELAN FUNGSIONAL

Model fungsional menggambarkan proses bisnis dan interaksi sistem informasi dengan lingkungannya. Dalam pengembangan sistem berorientasi objek, dua jenis model digunakan untuk menggambarkan fungsionalitas sistem informasi: use case dan diagram aktivitas. Use case digunakan untuk menggambarkan fungsi dasar dari sistem informasi. Diagram aktivitas mendukung pemodelan logis dari proses bisnis dan alur kerja. Keduanya dapat digunakan untuk menggambarkan sistem saat in saat ini dan sistem yang akan dikembangkan. Bab ini menjelaskan proses bisnis dan pemodelan fungsional sebagai sarana untuk mendokumentasikan dan memahami persyaratan dan untuk memahami perilaku fungsional atau eksternal sistem.

4.1 Tujuan

- Memahami proses yang digunakan untuk mengidentifikasi proses bisnis dan Use-Case.
- Memahami proses yang digunakan untuk membuat diagram use-case.
- Memahami proses yang digunakan untuk memodelkan proses bisnis dengan diagram aktivitas.
- Memahami aturan dan pedoman gaya untuk diagram aktivitas.
- Memahami proses yang digunakan untuk membuat deskripsi use-case.
- Pahami aturan dan pedoman gaya untuk deskripsi Use-Case.
- Mampu membuat model fungsional proses bisnis dengan menggunakan diagram use-case, diagram aktivitas, dan deskripsi use-case.

4.2 Pendahuluan

Bab sebelumnya membahas teknik pengumpulan kebutuhan yang populer, seperti wawancara, JAD, dan observasi. Dengan menggunakan teknik ini, analis menentukan persyaratan dan membuat definisi persyaratan. Definisi persyaratan mendefinisikan apa yang harus dilakukan sistem. Dalam bab ini, kita membahas bagaimana informasi yang dikumpulkan menggunakan teknik ini diatur dan disajikan dalam bentuk use-case dan diagram aktivitas dan deskripsi use-case. Karena Unified Modeling Language (UML) telah diterima sebagai notasi standar oleh Object Management Group (OMG), hampir semua proyek pengembangan berorientasi objek saat ini menggunakan model ini untuk mendokumentasikan dan mengatur persyaratan yang diperoleh selama alur kerja analisis.

Seperti yang ditunjukkan dalam Bab 1, semua pendekatan pengembangan sistem berorientasi objek didorong oleh Use-Case, arsitektur-sentris, dan iteratif dan inkremental. Use case adalah cara formal untuk merepresentasikan cara sistem bisnis berinteraksi dengan lingkungannya. Pada dasarnya, use case adalah gambaran tingkat tinggi dari proses bisnis dalam sistem informasi bisnis. Dari perspektif praktis, use case mewakili seluruh dasar untuk sistem berorientasi objek. Use case dapat mendokumentasikan sistem saat ini (yaitu, sistem apa adanya) atau sistem baru yang sedang dikembangkan (yaitu, sistem yang akan datang). Mengingat bahwa sistem berorientasi objek didorong oleh Use-Case, Use-Case juga membentuk dasar untuk pengujian (lihat Bab 12) dan desain antarmuka pengguna (lihat Bab 10). Dua bentuk use-case driven testing adalah walkthrough (dijelaskan nanti di bab ini) dan role-playing (dijelaskan di Bab 5).

Dari perspektif arsitektur-sentris, pemodelan use-case mendukung penciptaan pandangan eksternal atau fungsional dari proses bisnis yang menunjukkan bagaimana

pengguna melihat proses daripada mekanisme internal dimana proses dan sistem pendukung beroperasi. Pandangan berbasis arsitektur struktural dan perilaku dijelaskan dalam Bab 5 dan 6, masing-masing. Akhirnya, semua pendekatan pengembangan sistem berorientasi objek dikembangkan secara inkremental dan iteratif. Meskipun kami menyajikan tiga pandangan arsitektur secara berurutan, hal ini dilakukan terutama untuk alasan pedagogis. Anda akan menemukan bahwa Anda tidak hanya perlu melakukan iterasi di seluruh proses bisnis dan model fungsional (dijelaskan dalam bab ini), Anda juga harus melakukan iterasi di ketiga tampilan arsitektur untuk sepenuhnya menangkap dan mewakili persyaratan untuk sistem informasi bisnis.

Diagram aktivitas biasanya digunakan untuk menambah pemahaman kita tentang proses bisnis dan model Use-Case kita. Secara teknis, diagram aktivitas dapat digunakan untuk semua jenis aktivitas pemodelan proses. Dalam bab ini, kami menjelaskan penggunaannya dalam konteks pemodelan proses bisnis. Model proses menggambarkan bagaimana sistem bisnis beroperasi. Mereka menggambarkan proses atau aktivitas yang dilakukan dan bagaimana objek (data) bergerak di antara mereka. Model proses dapat digunakan untuk mendokumentasikan sistem saat ini (yaitu, sistem apa adanya) atau sistem baru yang sedang dikembangkan (yaitu, sistem yang akan datang), baik terkomputerisasi atau tidak. Banyak teknik pemodelan proses yang berbeda digunakan saat ini.

Diagram aktivitas dan Use-Case adalah model logis—model yang menggambarkan aktivitas domain bisnis tanpa menyarankan bagaimana aktivitas tersebut dilakukan. Model logis kadang-kadang disebut sebagai model domain masalah. Membaca use-case atau diagram aktivitas, pada prinsipnya, tidak boleh menunjukkan apakah suatu aktivitas terkomputerisasi atau manual, jika sepotong informasi dikumpulkan dalam bentuk kertas atau melalui Web, atau jika informasi ditempatkan di lemari arsip atau lemari besar. database. Detail fisik ini didefinisikan selama desain ketika model logis disempurnakan menjadi model fisik. Model-model ini memberikan informasi yang dibutuhkan untuk akhirnya membangun sistem. Dengan berfokus pada aktivitas logis terlebih dahulu, analis dapat fokus pada bagaimana bisnis harus berjalan tanpa terganggu dengan detail implementasi.

Sebagai langkah pertama, tim proyek mengumpulkan persyaratan dari pengguna (lihat Bab 3). Dengan menggunakan persyaratan yang dikumpulkan, tim proyek kemudian mengidentifikasi proses bisnis dan lingkungannya menggunakan use case dan diagram use case. Selanjutnya, pengguna bekerja sama dengan tim untuk memodelkan proses bisnis dalam bentuk diagram aktivitas, dan tim mendokumentasikan proses bisnis yang dijelaskan dalam Use-Case dan diagram aktivitas dengan membuat deskripsi Use-Case untuk setiap Use-Case. Terakhir, tim memverifikasi dan memvalidasi proses bisnis dengan memastikan bahwa ketiga model (diagram use-case, diagram aktivitas, dan deskripsi usecase) cocok satu sama lain. Setelah pemahaman saat ini tentang proses bisnis didokumentasikan dalam model fungsional, tim siap untuk beralih ke pemodelan struktural (lihat Bab 5).

Dalam bab ini, pertama-tama kami menjelaskan identifikasi proses bisnis menggunakan use case dan diagram use case. Kedua, kami menggambarkan pemodelan proses bisnis dengan diagram aktivitas. Ketiga, kami menjelaskan deskripsi use-case, elemennya, dan seperangkat pedoman untuk membuatnya. Keempat, kami menggambarkan proses verifikasi dan validasi proses bisnis dan model fungsional.

4.3 IDENTIFIKASI PROSES BISNIS DENGAN USE-CASE DAN DIAGRAM USE-CASE

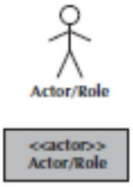
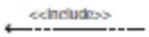
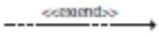
Pada bab sebelumnya, kita telah mempelajari tentang strategi dan teknik yang berguna dalam mengidentifikasi proses bisnis yang berbeda dari suatu sistem sehingga definisi persyaratan dapat dibuat. Pada bagian ini, kita belajar bagaimana memulai pemodelan proses

bisnis dengan use case dan diagram use case. Seorang analis dapat menggunakan Use-Case dan diagram Use-Case untuk lebih memahami fungsionalitas sistem pada tingkat yang sangat tinggi. Biasanya, karena diagram use-case menyediakan cara yang sederhana dan langsung untuk mengkomunikasikan kepada pengguna apa yang akan dilakukan sistem, diagram use-case digambar saat mengumpulkan dan mendefinisikan persyaratan untuk sistem. Dengan cara ini, diagram Use-Case dapat mendorong pengguna untuk menyediakan persyaratan tingkat tinggi tambahan. Diagram use-case menggambarkan dengan cara yang sangat sederhana fungsi utama sistem dan berbagai jenis pengguna yang akan berinteraksi dengannya. Gambar 4-1 menjelaskan aturan sintaks dasar untuk diagram use-case. Gambar 4-2 menyajikan diagram Use-Case untuk sistem janji temu kantor dokter yang diperkenalkan pada bab sebelumnya. Kita dapat melihat dari diagram bahwa pasien, dokter, dan personel manajemen akan menggunakan sistem janji temu masing-masing untuk mengelola janji temu, mencatat ketersediaan, dan membuat jadwal.

Elemen Diagram Use-Case

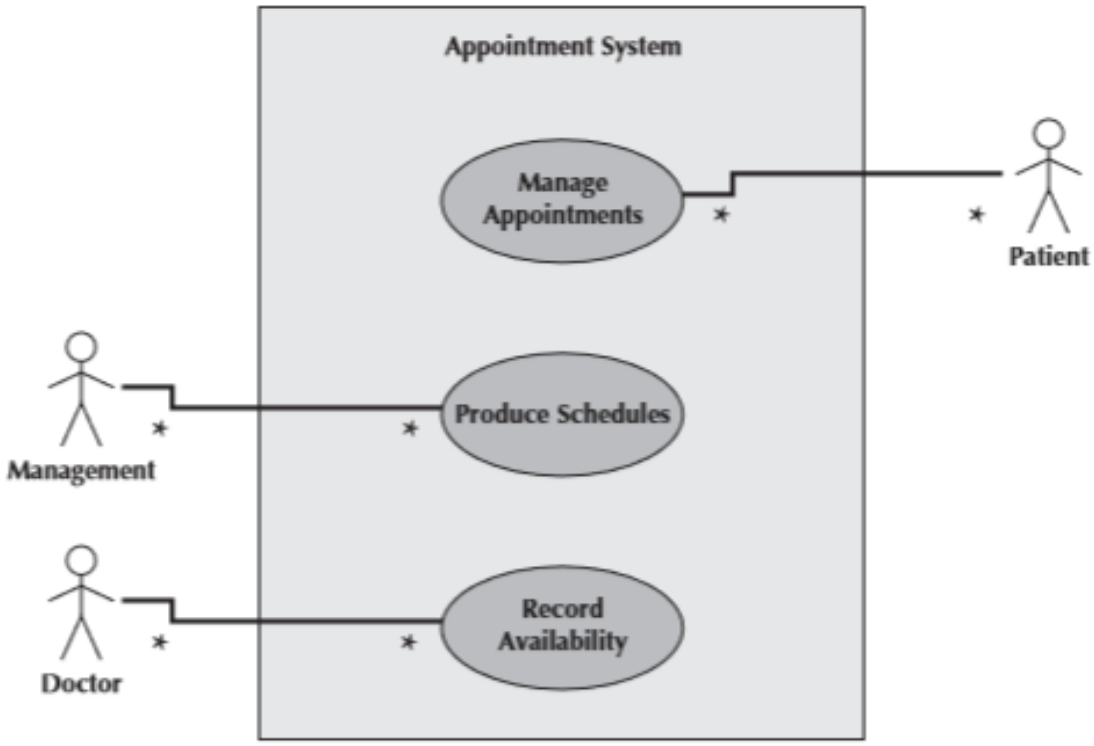
Elemen-elemen diagram use-case meliputi aktor, use case, batasan subjek, dan serangkaian hubungan antara aktor, aktor dan use case, dan use case. Hubungan tersebut terdiri dari hubungan asosiasi, include, extend, dan generalization. Masing-masing elemen ini dijelaskan selanjutnya.

Aktor. Figur tongkat pada diagram mewakili aktor (lihat Gambar 4-1). Aktor bukanlah pengguna tertentu melainkan peran yang dapat dimainkan pengguna saat berinteraksi dengan sistem. Seorang aktor juga dapat mewakili sistem lain di mana sistem saat ini berinteraksi. Dalam hal ini, aktor secara opsional dapat diwakili oleh persegi panjang yang berisi <<actor>> dan nama sistem. Pada dasarnya, aktor mewakili elemen utama dalam lingkungan di mana sistem beroperasi. Aktor dapat memberikan input ke sistem, menerima output dari sistem, atau keduanya. Diagram pada Gambar 4-2 menunjukkan bahwa tiga aktor akan berinteraksi dengan sistem penunjukan (pasien, dokter, dan manajemen).

Aktor: ■ Adalah orang atau sistem yang medapatkan peran di luar subjek. ■ Digambarkan sebagai figur tongkat (default) atau, jika aktor nonmanusia terlibat, persegi panjang dengan <<actor>> di dalamnya (alternatif). ■ Dilabeli dengan perannya. ■ Dapat dikaitkan dengan aktor lain menggunakan asosiasi spesialisasi/superclass, dilambangkan dengan panah dengan kepala panah bulat. ■ Ditempatkan di luar batas subjek.	
Use Case: ■ Merupakan bagian utama dari sistem fungsionalitas. ■ Dapat memperpanjang *use case* lain. ■ Dapat menyertakan *use case* lain. ■ Ditempatkan di dalam batas sistem. ■ Dilabeli dengan deskrispsi frase kata kerja-kata benda.	
Batas subjek: ■ Termasuk nama subjek di dalam atau di atas. ■ Mewakili ruang lingkup subjek, misalnya, sistem atau proses bisnis individu.	
Hubungan asosiasi: ■ Menghubungkan aktor dengan *use case* yang berinteraksi dengannya.	
Cakupan hubungan: ■ Mewakili penyertaan fungsionalitas satu *use case* dalam *use case* lainnya. ■ Memiliki panah yang ditarik dari *use case* dasar ke *use case* yang digunakan.	
Perpanjangan hubungan: ■ Merupakan perpanjangan dari *use case* untuk memasukkan perilaku opsional. ■ Memiliki panah yang ditarik dari *use case* ekstensi ke *use case* dasar.	
Hubungan generalisasi: ■ Mewakili *use case* khusus ke *case* yang lebih umum. ■ Memiliki panah yang ditarik dari *use case* khusus ke *use case* dasar.	

Gambar 4-1 Sintaks untuk Use-Case Diagram

Terkadang seorang aktor memainkan peran khusus dari tipe aktor yang lebih umum. Misalnya, mungkin ada saat-saat ketika pasien baru berinteraksi dengan sistem dengan cara yang agak berbeda dari pasien umum. Dalam hal ini, aktor khusus (yaitu, pasien baru) dapat ditempatkan pada model, ditunjukkan menggunakan garis dengan segitiga berongga di ujung aktor yang lebih umum (yaitu, pasien). Aktor khusus mewarisi perilaku aktor yang lebih umum dan memperluasnya dalam beberapa cara (lihat Gambar 4-3).



Gambar 4-2 Use-Case Diagram untuk Sistem Penunjukan

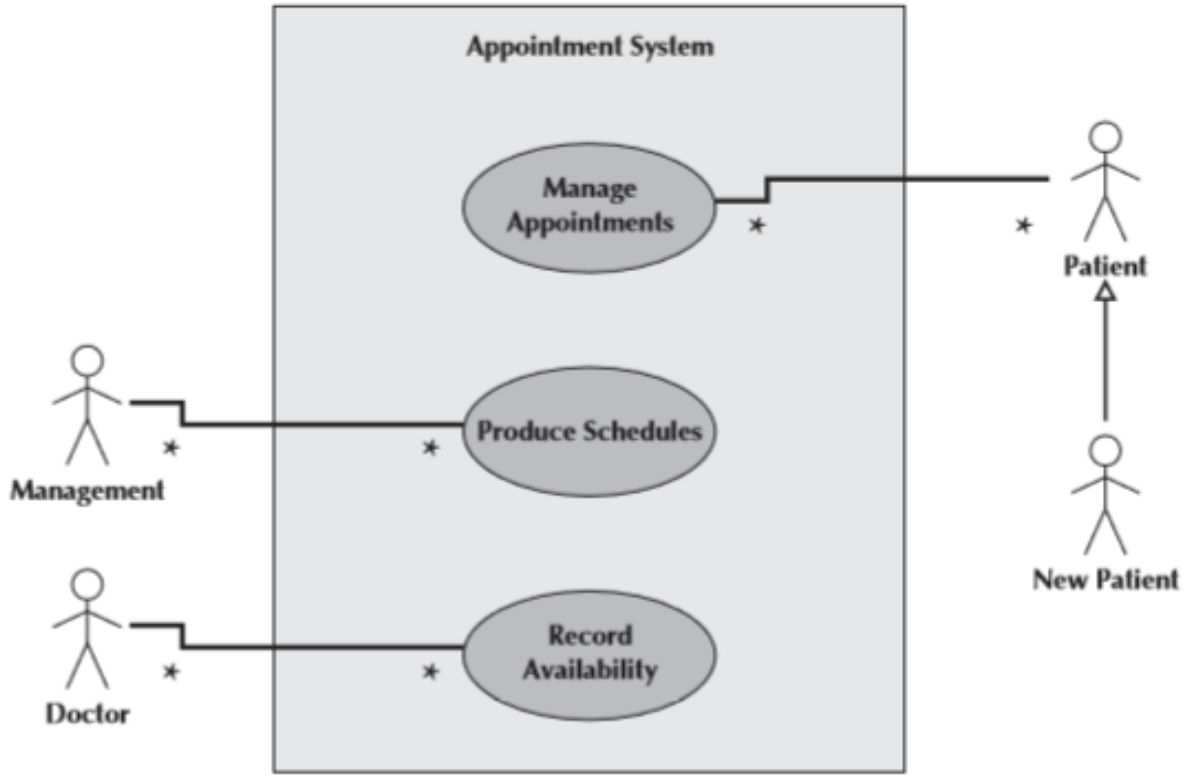
Use-Case Asosiasi terhubung ke aktor melalui hubungan asosiasi; hubungan ini menunjukkan dengan Use-Case mana aktor berinteraksi (lihat Gambar 4-1). Garis yang ditarik dari aktor ke use case menggambarkan sebuah asosiasi. Asosiasi biasanya mewakili komunikasi dua arah antara use case dan aktor. Jika komunikasi hanya satu arah, maka panah yang kokoh dapat digunakan untuk menunjukkan arah arus informasi.

Misalnya, pada Gambar 4-2, aktor Pasien berkomunikasi dengan Use-Case Kelola Janji Temu. Karena tidak ada panah di asosiasi, komunikasinya dua arah. Akhirnya, adalah mungkin untuk mewakili multiplisitas asosiasi. Gambar 4-2 menunjukkan tanda bintang (*) di kedua ujung hubungan antara Pasien dan Use-Case Kelola Janji Temu. Ini hanya menunjukkan bahwa pasien individu (contoh dari aktor Pasien) mengeksekusi Use-Case Kelola Janji Temu sebanyak yang dia inginkan dan mungkin untuk bagian penunjukan dari Use-Case Kelola Janji Temu untuk dieksekusi oleh banyak pasien yang berbeda. . Dalam kebanyakan kasus, jenis hubungan banyak-ke-banyak ini sesuai. Namun, dimungkinkan untuk membatasi jumlah pasien yang dapat dikaitkan dengan Use-Case Kelola Janji Temu. Kami membahas masalah multiplisitas secara rinci dalam bab berikutnya sehubungan dengan diagram kelas.

Use Case. Sebuah use case, digambarkan dengan oval di UML, adalah proses utama yang dilakukan sistem dan menguntungkan aktor atau aktor dalam beberapa cara (lihat Gambar 4-1); itu diberi label menggunakan frase kata kerja-kata benda deskriptif. Kita dapat mengetahui dari Gambar 4-2 bahwa sistem memiliki tiga Use-Case utama: Kelola Janji Temu, Jadwal Produksi, dan Ketersediaan Rekaman.

Ada kalanya use case menyertakan, memperluas, atau menggeneralisasi fungsionalitas use case lain dalam diagram. Ini ditunjukkan dengan menggunakan hubungan include, extend, dan generalization. Untuk meningkatkan kemudahan memahami diagram use-case, use case tingkat tinggi biasanya digambar di atas yang lebih rendah. Mungkin lebih mudah untuk memahami hubungan ini dengan bantuan contoh. Mari kita asumsikan bahwa setiap kali

pasien membuat janji, pasien diminta untuk memverifikasi pengaturan pembayaran. Namun, kadang-kadang perlu untuk benar-benar membuat pengaturan pembayaran baru. Oleh karena itu, kami mungkin ingin memiliki Use-Case yang disebut Lakukan Pengaturan Pembayaran yang memperluas Use-Case Kelola Janji Temu untuk menyertakan fungsionalitas tambahan ini. Pada Gambar 4-4, panah berlabel perpanjangan ditarik dari use case Make Payment Arrangements ke use case Manage Appointment untuk menunjukkan hubungan use-case khusus ini. Use-Case Lakukan Pengaturan Pembayaran digambar lebih rendah daripada Use-Case Kelola Janji Temu.



Gambar 4-3 Use-Case Diagram dengan Aktor Khusus

Demikian pula, ada kalanya satu use case berisi fungsi umum yang digunakan oleh use case lain. Misalnya, ada use case yang disebut Manage Schedule yang melakukan beberapa tugas rutin yang diperlukan untuk mempertahankan jadwal janji temu kantor dokter, dan dua use case Record Availability dan Produce Schedule keduanya melakukan tugas rutin. Gambar 4-4 menunjukkan bagaimana kita dapat merancang sistem sehingga Kelola Jadwal adalah Use-Case bersama yang digunakan oleh orang lain. Panah berlabel include digunakan untuk menunjukkan hubungan include, dan use case yang disertakan digambar di bawah use case yang mengandungnya. Perhatikan bahwa panah diambil dari Use-Case Ketersediaan Rekaman dan Jadwal Produksi ke Use-Case Kelola Jadwal yang umum.

Akhirnya, ada kalanya masuk akal untuk menggunakan hubungan generalisasi untuk menyederhanakan Use-Case individu. Misalnya pada Gambar 4-4, use case Kelola Janji Temu telah dispesialisasikan untuk menyertakan use case untuk Pasien Lama dan Pasien Baru. Use-Case Buat Aplikasi Pasien Lama mewarisi fungsionalitas Use-Case Kelola Janji Temu (termasuk ekstensi Use-Case Lakukan Pengaturan Pembayaran) dan memperluas fungsinya sendiri dengan Use-Case Perbarui Informasi Pasien. Use-Case Buat Aplikasi Pasien Baru juga mewarisi semua fungsi Use-Case Kelola Janji Temu generik dan memanggil Use-Case Buat Pasien Baru, yang mencakup fungsionalitas yang diperlukan untuk memasukkan pasien baru ke dalam database pasien. Hubungan generalisasi direpresentasikan sebagai panah berongga

yang tidak berlabel dengan use case yang lebih umum lebih tinggi daripada use case yang lebih rendah. Juga, perhatikan bahwa kami telah menambahkan aktor khusus kedua, Pasien Lama, dan bahwa aktor Pasien sekarang hanyalah generalisasi dari aktor Pasien Lama dan Baru.

Batas Subyek. Use-Case diapit dalam batas subjek, yang merupakan kotak yang mendefinisikan ruang lingkup sistem dan dengan jelas menggambarkan bagian mana dari diagram yang eksternal atau internal (lihat Gambar 4-1). Salah satu keputusan yang lebih sulit untuk dibuat adalah di mana menggambar batas subjek. Batas subjek dapat digunakan untuk memisahkan sistem perangkat lunak dari lingkungannya, subsistem dari subsistem lain dalam sistem perangkat lunak, atau proses individu dalam sistem perangkat lunak. Mereka juga dapat digunakan untuk memisahkan sistem informasi, termasuk perangkat lunak dan aktor internal, dari lingkungannya. Perawatan harus diambil untuk memutuskan apa ruang lingkup sistem informasi yang akan.

Nama subjek dapat muncul di dalam atau di atas kotak. Batas subjek digambar berdasarkan ruang lingkup sistem. Dalam sistem pengangkatan, kami berasumsi bahwa aktor Manajemen dan Doktor berada di luar cakupan sistem; yaitu, mereka menggunakan sistem. Kita bisa saja memasukkan resepsionis sebagai aktor. Namun, dalam kasus ini, kami berasumsi bahwa resepsionis adalah aktor internal yang merupakan bagian dari Use-Case Kelola Janji Temu yang berinteraksi dengan aktor Pasien. Oleh karena itu, resepsionis tidak digambarkan pada diagram.

Mengidentifikasi Use-Case Utama

Langkah pertama adalah meninjau definisi persyaratan (lihat Gambar 3-1). Ini membantu analis untuk mendapatkan gambaran lengkap tentang proses bisnis yang mendasari yang dimodelkan.

Langkah kedua adalah mengidentifikasi batasan subjek. Ini membantu analis untuk mengidentifikasi ruang lingkup sistem. Namun, saat kami bekerja melalui proses pengembangan, batas sistem kemungkinan besar akan berubah.

Langkah ketiga adalah mengidentifikasi aktor utama dan tujuan mereka. Aktor utama yang terlibat dengan sistem berasal dari daftar pemegang kepentingan dan pengguna. Ingatlah bahwa pemegang kepentingan adalah orang, kelompok, atau organisasi yang dapat mempengaruhi (atau akan terpengaruh oleh) sistem baru, sedangkan aktor adalah peran yang dimainkan oleh pemegang kepentingan atau pengguna, bukan pengguna tertentu (misalnya, dokter, bukan pengguna). Dr.Jones). Sasaran mewakili fungsionalitas yang harus disediakan sistem untuk aktor agar sistem berhasil. Mengidentifikasi tugas-tugas yang harus dilakukan setiap aktor dapat memfasilitasi hal ini. Misalnya, apakah aktor perlu membuat, membaca, memperbarui, menghapus, atau mengeksekusi (CRUDE) informasi apa pun yang saat ini ada di sistem, apakah ada perubahan eksternal yang harus diinformasikan oleh aktor ke sistem, atau adakah informasi yang sistem harus memberikan aktor? Langkah 2 dan 3 saling terkait. Ketika aktor diidentifikasi dan tujuan mereka terungkap, batas sistem akan berubah.

Langkah keempat adalah mengidentifikasi proses bisnis dan Use-Case utama. Daripada melompat ke satu Use-Case dan menggambarkannya sepenuhnya pada saat ini, kami hanya ingin mengidentifikasi Use-Case. Mengidentifikasi hanya Use-Case utama saat ini mencegah pengguna dan analis melupakan proses bisnis utama dan membantu pengguna menjelaskan keseluruhan rangkaian proses bisnis yang menjadi tanggung jawab mereka. Penting pada titik ini untuk memahami dan mendefinisikan akronim dan jargon sehingga tim proyek dan orang lain dari luar grup pengguna dapat dengan jelas memahami Use-Case. Sekali lagi, definisi persyaratan adalah titik awal yang sangat berguna untuk langkah ini.

Langkah kelima adalah dengan hati-hati meninjau set Use-Case saat ini. Mungkin perlu untuk membagi beberapa dari mereka menjadi beberapa Use-Case atau menggabungkan beberapa dari mereka menjadi satu Use-Case. Juga, berdasarkan set saat ini, Use-Case baru dapat diidentifikasi. Anda harus ingat bahwa mengidentifikasi use case adalah proses berulang, dengan pengguna sering berubah pikiran tentang apa itu use case dan apa yang termasuk di dalamnya. Sangat mudah untuk terjebak dalam detail pada saat ini, jadi Anda harus ingat bahwa tujuan pada langkah ini adalah hanya mengidentifikasi Use-Case utama. Misalnya, dalam contoh kantor dokter pada Gambar 4-2, kami mendefinisikan satu Use-Case sebagai Kelola Janji Temu. Use-Case ini mencakup kasus untuk pasien baru dan pasien lama, serta saat pasien mengubah atau membatalkan janji temu. Kita dapat mendefinisikan setiap aktivitas ini (membuat janji temu, mengubah janji temu, atau membatalkan janji temu) sebagai Use-Case yang terpisah, tetapi ini akan menciptakan sekumpulan besar Use-Case kecil.

Triknya adalah memilih ukuran yang tepat sehingga Anda mendapatkan tiga hingga sembilan Use-Case di setiap sistem. Jika tim proyek menemukan lebih dari delapan Use-Case, ini menunjukkan bahwa Use-Case terlalu kecil atau batas sistem terlalu besar. Jika ada lebih dari sembilan Use-Case, Use-Case harus dikelompokkan bersama ke dalam paket (yaitu, kelompok logis dari Use-Case) untuk membuat diagram lebih mudah dibaca dan menjaga model pada tingkat kompleksitas yang wajar. Sederhana pada saat itu untuk mengurutkan Use-Case dan mengelompokkan Use-Case kecil ini menjadi Use-Case yang lebih besar yang mencakup beberapa kasus kecil atau untuk mengubah batas sistem.

Membuat Use-Case

Diagram Pada dasarnya, menggambar diagram use-case sangat mudah setelah use case dirinci. Diagram Use-Case yang sebenarnya mendorong penggunaan penyembunyian informasi. Satu-satunya bagian yang digambarkan pada diagram use-case adalah batas sistem, use case itu sendiri, aktor, dan berbagai asosiasi antara komponen-komponen ini. Kekuatan utama dari diagram use-case adalah bahwa ia menyediakan pengguna dengan gambaran umum dari proses bisnis. Namun, ingat bahwa setiap kali use case berubah, hal itu dapat mempengaruhi diagram use case. Ada empat langkah utama dalam menggambar diagram use-case.

Pertama, kita menempatkan dan menggambar use case pada diagram. Ini diambil langsung dari Use-Case utama yang diidentifikasi sebelumnya. Asosiasi Use-Case khusus (termasuk, memperluas, atau generalisasi) juga ditambahkan ke model pada titik ini. Hati-hati dalam menyusun diagram. Tidak ada urutan formal untuk use case, sehingga dapat ditempatkan dengan cara apa pun yang diperlukan untuk membuat diagram mudah dibaca dan untuk meminimalkan jumlah garis yang bersilangan. Seringkali perlu menggambar ulang diagram beberapa kali dengan Use-Case di tempat yang berbeda untuk membuat diagram mudah dibaca. Juga, untuk tujuan pemahaman, tidak boleh ada lebih dari tiga hingga sembilan Use-Case pada Use-Case penghitungan model yang telah difaktorkan dan sekarang dikaitkan dengan Use-Case lain melalui hubungan penyertaan, perluasan, atau generalisasi.

Kedua, aktor ditempatkan dan digambar pada diagram. Untuk meminimalkan jumlah garis yang bersilangan pada diagram, aktor harus ditempatkan di dekat use case yang terkait dengannya.

Ketiga, batas subjek ditarik. Ini membentuk batas subjek, memisahkan Use-Case (yaitu, fungsionalitas subjek) dari aktor (yaitu, peran pengguna eksternal).

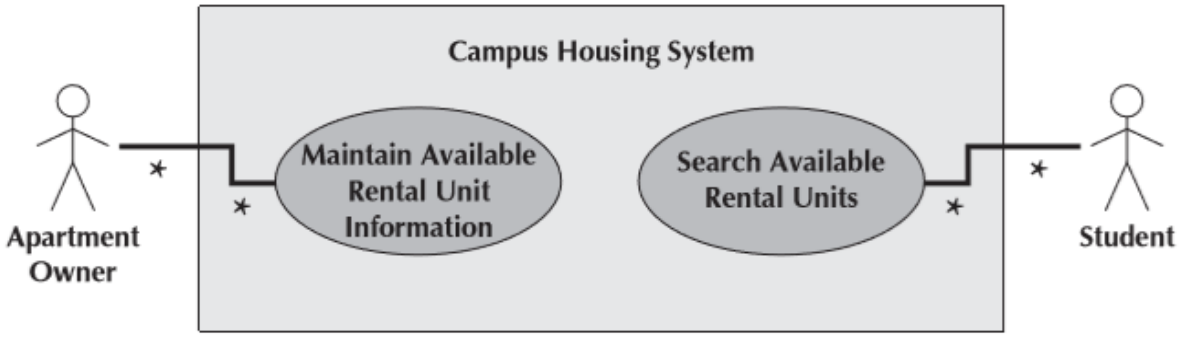
Langkah keempat dan terakhir adalah menambahkan asosiasi dengan menggambar garis untuk menghubungkan aktor ke use case tempat mereka berinteraksi. Tidak ada urutan yang tersirat dalam diagram, dan item yang ditambahkan di sepanjang jalan tidak harus

ditempatkan dalam urutan tertentu; oleh karena itu, mungkin membantu untuk mengatur ulang simbol sedikit untuk meminimalkan jumlah garis yang bersilangan, membuat diagram tidak terlalu membingungkan.

Contoh Perumahan Kampus. Identifikasi aktor dan Use-Case utama untuk proses bisnis tingkat tinggi berikut dalam sistem perumahan yang dijalankan oleh layanan perumahan kampus. Layanan perumahan kampus membantu mahasiswa menemukan apartemen. Pemilik apartemen melengkapi formulir informasi tentang unit sewa yang tersedia (misalnya, lokasi, jumlah kamar tidur, sewa bulanan), yang kemudian dimasukkan ke dalam database. Siswa dapat mencari database ini melalui Web untuk menemukan apartemen yang memenuhi kebutuhan mereka (misalnya, apartemen dua kamar tidur seharga Rp.6.000.000 atau kurang per bulan dalam jarak setengah mil dari kampus) dan menghubungi pemilik apartemen secara langsung untuk melihat apartemen dan mungkin menyewanya . Pemilik apartemen menghubungi layanan untuk menghapus daftar mereka ketika mereka telah menyewa apartemen mereka.

Sebagai langkah pertama, kami mengidentifikasi aktor utama, proses bisnis utama, dan Use-Case utama. Dalam hal ini, pelaku utama adalah pemilik apartemen dan mahasiswa. Tujuan dari pelaku utama adalah kedua sisi transaksi sewa, yaitu menyewakan apartemen. Proses bisnis utama dan Use-Case untuk memungkinkan para pelaku mewujudkan tujuan mereka adalah untuk menjaga informasi unit sewa yang tersedia untuk pemilik apartemen dan untuk menemukan unit sewa yang sesuai untuk dipertimbangkan bagi siswa. Menggunakan aktor yang diidentifikasi dan Use-Case dan mengikuti proses yang dijelaskan di atas, diagram Use-Case pada Gambar 4-5 dibuat. Perhatikan bahwa diagram hanya mencakup dua Use-Case dan dua aktor. Dalam hal ini, Use-Case Pertahankan Informasi Unit Sewa yang Tersedia sebenarnya mencakup dua subproses terpisah. Pemilik apartemen dapat menambahkan unit sewa yang telah tersedia, dan dapat menghapus unit sewa yang telah disewa dan tidak tersedia lagi. Seorang siswa dapat mencari Use-Case Cari Unit Sewa yang tersedia dengan menggunakan tiga kriteria terpisah: jarak dari kampus, jumlah kamar tidur, dan sewa bulanan. Kriteria ini dapat digunakan secara individual atau dengan kombinasi dari ketiganya. Kami akan kembali ke contoh ini di bagian berikutnya dari bab ini. Namun, sebelum kita melanjutkan, selanjutnya kita akan menjelaskan sistem yang sedikit lebih terlibat untuk perpustakaan universitas.

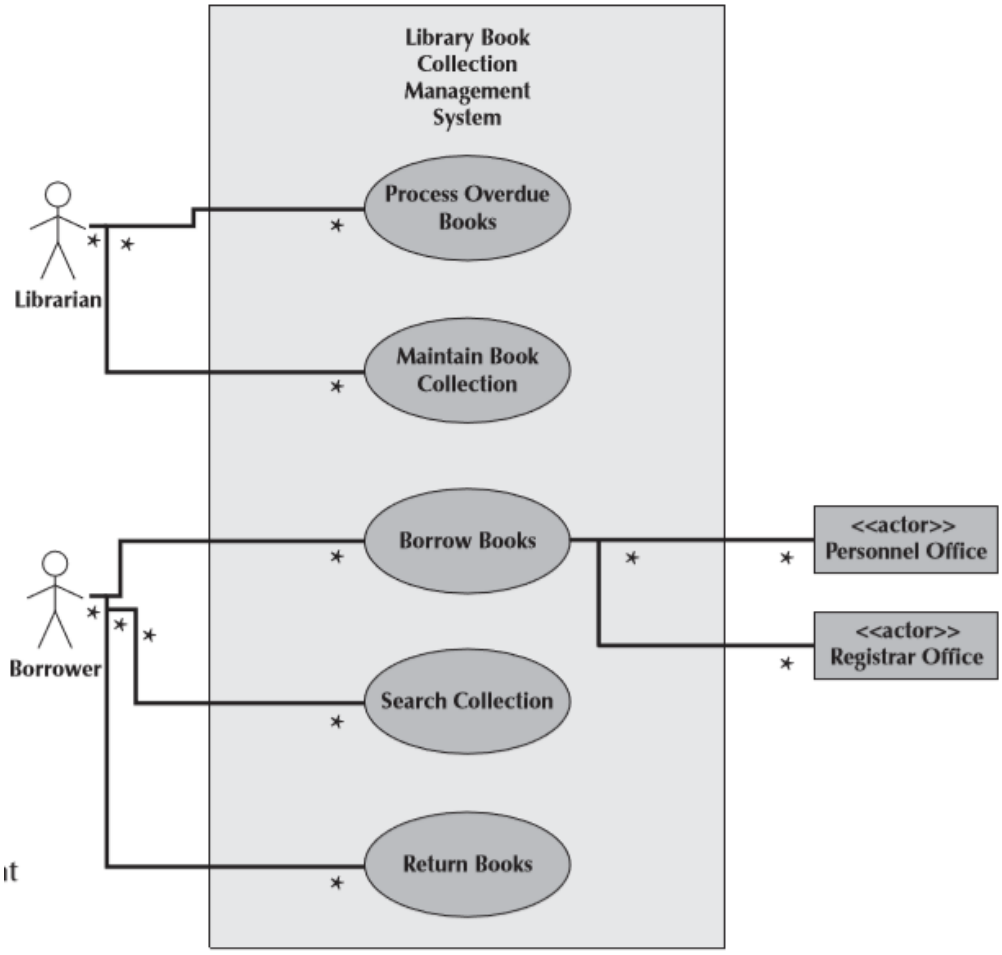
Contoh Perpustakaan. Persyaratan fungsional untuk sistem sirkulasi perpustakaan universitas otomatis mencakup kebutuhan untuk mendukung kegiatan pencarian, peminjaman, dan pemeliharaan buku. Sistem harus mendukung pencarian berdasarkan judul, penulis, kata utama, dan ISBN. Pencarian database koleksi perpustakaan harus tersedia di terminal di perpustakaan dan tersedia untuk calon peminjam melalui Web. Jika buku yang diminati saat ini sedang diperiksa, peminjam yang sah harus diizinkan untuk meminta buku itu dikembalikan. Setelah buku diperiksa kembali, peminjam yang meminta buku harus diberitahu tentang ketersediaan buku.



Gambar 4-5 Diagram Kasus Perumahan Kampus

Kegiatan peminjaman dibangun di sekitar memeriksa buku dan mengembalikan buku oleh peminjam. Ada tiga jenis peminjam: mahasiswa, dosen atau staf, dan tamu. Terlepas dari jenis peminjam, peminjam harus memiliki KTP yang masih berlaku. Jika peminjam adalah mahasiswa, sistem akan memeriksa dengan database mahasiswa pendaftar memvalidasi kartu ID. Jika peminjam adalah staf pengajar atau staf, sistem memeriksa dengan database karyawan kantor personalia memvalidasi kartu identitas. Jika peminjam adalah tamu, kartu ID diperiksa terhadap database peminjam perpustakaan itu sendiri. Jika KTP tersebut valid, sistem juga harus memeriksa untuk menentukan apakah peminjam memiliki buku yang telah jatuh tempo atau denda yang belum dibayar. Jika KTP tidak berlaku, peminjam memiliki buku yang jatuh tempo, atau peminjam memiliki denda yang belum dibayar, sistem harus menolak permintaan peminjam untuk memeriksa buku, jika tidak, permintaan peminjam harus dipenuhi. Jika sebuah buku di-check out, sistem harus memperbarui database koleksi perpustakaan untuk mencerminkan status baru buku tersebut.

Kegiatan pemeliharaan buku berkaitan dengan penambahan dan penghapusan buku dari koleksi buku perpustakaan. Ini membutuhkan manajer perpustakaan untuk menambah dan menghapus buku secara logis dan fisik. Buku yang dibeli oleh perpustakaan atau buku yang dikembalikan dalam keadaan rusak biasanya menyebabkan aktivitas ini. Jika sebuah buku dipastikan rusak saat dikembalikan dan perlu dikeluarkan dari koleksi, peminjam terakhir akan dikenakan denda. Namun, jika buku dapat diperbaiki, tergantung pada biaya perbaikan, peminjam mungkin tidak akan dikenakan denda. Setiap hari Senin, perpustakaan mengirimkan e-mail pengingat kepada peminjam yang memiliki buku terlambat. Jika sebuah buku terlambat lebih dari dua minggu, peminjam dinilai denda. Tergantung pada berapa lama buku tersebut jatuh tempo, peminjam dapat dikenakan denda tambahan setiap hari Senin.



Gambar 4-6 Diagram Use-Case Sistem Pengelolaan Koleksi Buku Perpustakaan

Untuk memulai, kita perlu mengidentifikasi Use-Case utama dan membuat diagram Use-Case yang mewakili proses bisnis tingkat tinggi dalam situasi bisnis yang baru saja dijelaskan. Berdasarkan langkah-langkah untuk mengidentifikasi Use-Case utama, kita perlu meninjau definisi persyaratan dan mengidentifikasi batasan (lingkup) masalah. Berdasarkan uraian masalah tersebut, terlihat bahwa sistem yang akan dibuat hanya sebatas pengelolaan koleksi buku perpustakaan. Hal selanjutnya yang perlu kita lakukan adalah mengidentifikasi aktor utama dan proses bisnis yang perlu didukung oleh sistem. Berdasarkan kebutuhan fungsional yang diuraikan, pelaku utama adalah peminjam dan pustakawan, sedangkan proses bisnis utama adalah peminjaman buku, pengembalian buku, pencarian koleksi buku, pemeliharaan koleksi buku, dan pemrosesan buku yang telah jatuh tempo. Sekarang kita telah mengidentifikasi semua aktor dan Use-Case utama, kita dapat menggambar diagram Use-Case yang mewakili gambaran umum sistem manajemen koleksi buku perpustakaan (lihat Gambar 4-6). Perhatikan penambahan dua aktor bukan manusia (Kantor Personalia dan Kantor Panitera).

4.4 PEMODELAN PROSES BISNIS DENGAN DIAGRAM AKTIVITAS

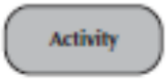
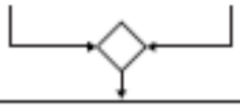
Model proses bisnis menggambarkan berbagai aktivitas yang, ketika digabungkan, mendukung proses bisnis. Proses bisnis biasanya melintasi departemen fungsional (misalnya, pembuatan produk baru melibatkan banyak aktivitas berbeda yang menggabungkan upaya banyak karyawan di banyak departemen). Dari perspektif berorientasi objek, proses ini melintasi beberapa objek. Banyak pendekatan pengembangan sistem berorientasi objek sebelumnya cenderung mengabaikan pemodelan proses bisnis. Namun, hari ini kami menyadari bahwa pemodelan proses bisnis itu sendiri adalah aktivitas yang sangat konstruktif yang dapat digunakan untuk memahami persyaratan yang dikumpulkan (lihat Bab 3). Salah satu masalah potensial dalam membangun model proses bisnis, dari perspektif pengembangan sistem berorientasi objek, adalah bahwa model tersebut cenderung memperkuat pola pikir dekomposisi fungsional. Namun, selama mereka digunakan dengan benar, model proses bisnis adalah alat yang sangat kuat untuk mengkomunikasikan pemahaman analis saat ini tentang persyaratan kepada pengguna.

Martin Schedlbauer menyediakan serangkaian praktik terbaik untuk diikuti saat memodelkan proses bisnis.

- Bersikaplah realistis, karena hampir tidak mungkin untuk mengidentifikasi segala sesuatu yang termasuk dalam proses bisnis pada titik ini dalam evolusi sistem. Bahkan jika kita bisa mengidentifikasi semuanya, semuanya tidak sama pentingnya.
- Jadilah tangkas karena meskipun kami mungkin tidak mengidentifikasi setiap fitur dari proses bisnis, fitur yang kami identifikasi harus diidentifikasi dengan cara yang ketat.
- Semua modeling adalah kegiatan kolaboratif/sosial. Oleh karena itu, pemodelan proses bisnis harus dilakukan dengan tim, bukan individu. Ketika seorang individu menciptakan sebuah model, kemungkinan mencampuradukkan atau menghilangkan tugas-tugas penting sangat meningkat.
- Jangan gunakan CASE tool untuk melakukan pemodelan tetapi gunakan papan tulis sebagai gantinya. Namun, setelah prosesnya dipahami, adalah ide yang baik untuk menggunakan CASE tool untuk mendokumentasikan proses tersebut.
- Pemodelan proses harus dilakukan secara iteratif. Saat Anda lebih memahami proses bisnis, Anda harus kembali ke versi proses yang terdokumentasi dan merevisinya.
- Saat memodelkan proses bisnis, tetap fokus pada proses spesifik itu. Jika tugas-tugas yang terkait dengan proses bisnis lain diidentifikasi, cukup catat tugas tersebut pada daftar tugas dan kembali ke proses bisnis yang sedang Anda modelkan.

- Ingatlah bahwa model proses bisnis adalah abstraksi dari kenyataan. Maksud kami, Anda tidak boleh memasukkan setiap tugas kecil dalam deskripsi proses bisnis saat ini. Ingat, Anda tidak bisa melupakan hutan pepatah demi pemahaman terperinci tentang satu pohon. Terlalu banyak detail pada saat ini dalam evolusi sistem dapat menyebabkan kebingungan dan benar-benar mencegah Anda memecahkan masalah mendasar yang ditangani oleh sistem baru.

Diagram aktivitas digunakan untuk memodelkan perilaku dalam proses bisnis yang tidak bergantung pada objek. Diagram aktivitas dapat digunakan untuk memodelkan semuanya, mulai dari alur kerja bisnis tingkat tinggi yang melibatkan banyak Use-Case berbeda, hingga detail Use-Case individual, hingga detail spesifik metode individual. Singkatnya, diagram aktivitas dapat digunakan untuk memodelkan semua jenis proses. Dalam bab ini, kami membatasi cakupan diagram aktivitas kami untuk mendokumentasikan dan memodelkan proses bisnis tingkat tinggi.

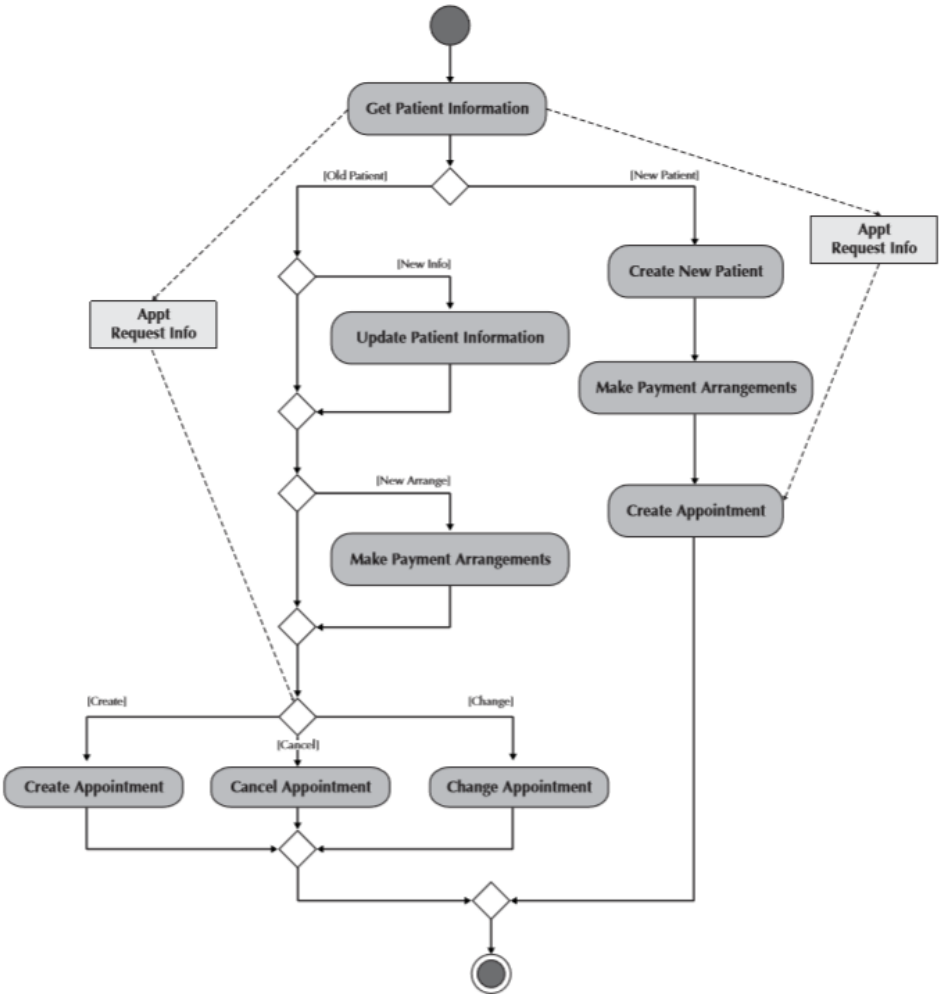
Action: ■ Sederhananya, bagian dari perilaku yang tidak dapat dikomposisi. ■ Diberi label dengan namanya.	
Activity: ■ Digunakan untuk mewakili serangkaian tindakan. ■ Diberi label dengan namanya.	
Simpul Objek ■ Digunakan untuk merepresentasikan sebuah objek yang terhubung dengan sekumpulan objek mengalir. ■ Diberi label dengan nama kelasnya.	
Aliran kontrol: ■ Menunjukkan urutan eksekusi.	
Aliran objek: ■ Menunjukkan aliran suatu objek dari satu aktivitas (atau tindakan) ke aktivitas (atau tindakan) lain.	
Simpul awal: ■ Menggambarkan awal dari serangkaian tindakan atau kegiatan.	
Simpul akhir aktivitas: ■ Digunakan untuk menghentikan semua aliran kontrol dan aliran objek dalam suatu aktivitas (atau tindakan).	
Simpul akhir aliran: ■ Digunakan untuk menghentikan aliran kontrol tertentu atau aliran objek.	
Simpul keputusan: ■ Digunakan untuk mewakili kondisi pengujian untuk memastikan bahwa aliran kontrol atau aliran objek hanya turun satu jalur. ■ Dilabeli dengan kriteria keputusan untuk melanjutkan ke jalur tertentu.	
Simpul gabungan: ■ Digunakan untuk menyatukan kembali jalur keputusan yang berbeda yang dibuat menggunakan simpul keputusan.	
Simpul cabang: ■ Digunakan untuk membagi perilaku menjadi serangkaian aktivitas (atau tindakan) paralel atau bersamaan.	
Simpul berhubungan: ■ Digunakan untuk menyatukan kembali serangkaian aktivitas (atau tindakan) paralel atau bersamaan.	
Jalur renang: ■ Digunakan untuk memecah diagram aktivitas menjadi baris dan kolom untuk menetapkan aktivitas individu (atau tindakan) kepada individu atau objek yang bertanggung jawab untuk melaksanakan aktivitas (atau tindakan). ■ Diberi label dengan nama individu atau objek yang bertanggung jawab.	

Gambar 4-7 Sintaks untuk Diagram Aktivitas

Elemen Diagram Aktivitas

Diagram aktivitas menggambarkan aktivitas utama dan hubungan antara aktivitas dalam suatu proses. Gambar 4-7 menunjukkan sintaks dari diagram aktivitas. Gambar 4-8 disajikan sebagai diagram aktivitas sederhana yang mewakili Use-Case Kelola Janji Temu dari sistem janji temu untuk contoh kantor dokter.

Tindakan dan Aktivitas. Tindakan dan aktivitas dilakukan untuk beberapa alasan bisnis tertentu. Tindakan dan aktivitas dapat mewakili perilaku manual atau terkomputerisasi. Mereka digambarkan dalam diagram aktivitas sebagai persegi panjang bulat (lihat Gambar 4-7). Mereka harus memiliki nama yang dimulai dengan kata kerja dan diakhiri dengan kata benda (mis., Dapatkan Informasi Pasien atau Lakukan Pengaturan Pembayaran). Nama harus pendek, namun berisi informasi yang cukup sehingga pembaca dapat dengan mudah memahami apa yang mereka lakukan. Satu-satunya perbedaan antara tindakan dan aktivitas adalah bahwa aktivitas dapat diuraikan lebih lanjut menjadi serangkaian aktivitas dan/atau tindakan, sedangkan tindakan mewakili bagian sederhana yang tidak dapat diuraikan dari keseluruhan perilaku yang dimodelkan. Biasanya, hanya aktivitas yang digunakan untuk proses bisnis atau pemodelan alur kerja. Dalam kebanyakan kasus, setiap aktivitas dikaitkan dengan use case. Diagram aktivitas pada Gambar 4-8 menunjukkan serangkaian aktivitas terpisah namun terkait untuk Use-Case Kelola Janji Temu (lihat Gambar 4-2, 4-3, dan 4-4): Dapatkan Informasi Pasien, Perbarui Informasi Pasien, Buat Pasien Baru , Buat Pengaturan Pembayaran, Buat Janji Baru, Ubah Janji, dan Batalkan Janji. Perhatikan bahwa aktivitas Buat Pengaturan Pembayaran dan Buat Janji Baru muncul dua kali dalam diagram: satu kali untuk pasien “lama” dan satu kali untuk pasien “baru”.



Gambar 4-8 Diagram Aktivitas untuk Use Case Kelola Janji Temu

Node Objek. Aktivitas dan tindakan biasanya memodifikasi atau mengubah objek. Node objek memodelkan objek-objek ini dalam diagram aktivitas. Node objek digambarkan dalam diagram aktivitas sebagai persegi panjang (lihat Gambar 4-7). Nama kelas objek ditulis di dalam persegi panjang. Pada dasarnya, node objek mewakili aliran informasi dari satu aktivitas ke aktivitas lain. Sistem penunjukan sederhana yang digambarkan pada Gambar 4-8 menunjukkan node objek yang mengalir dari aktivitas Dapatkan Informasi Pasien.

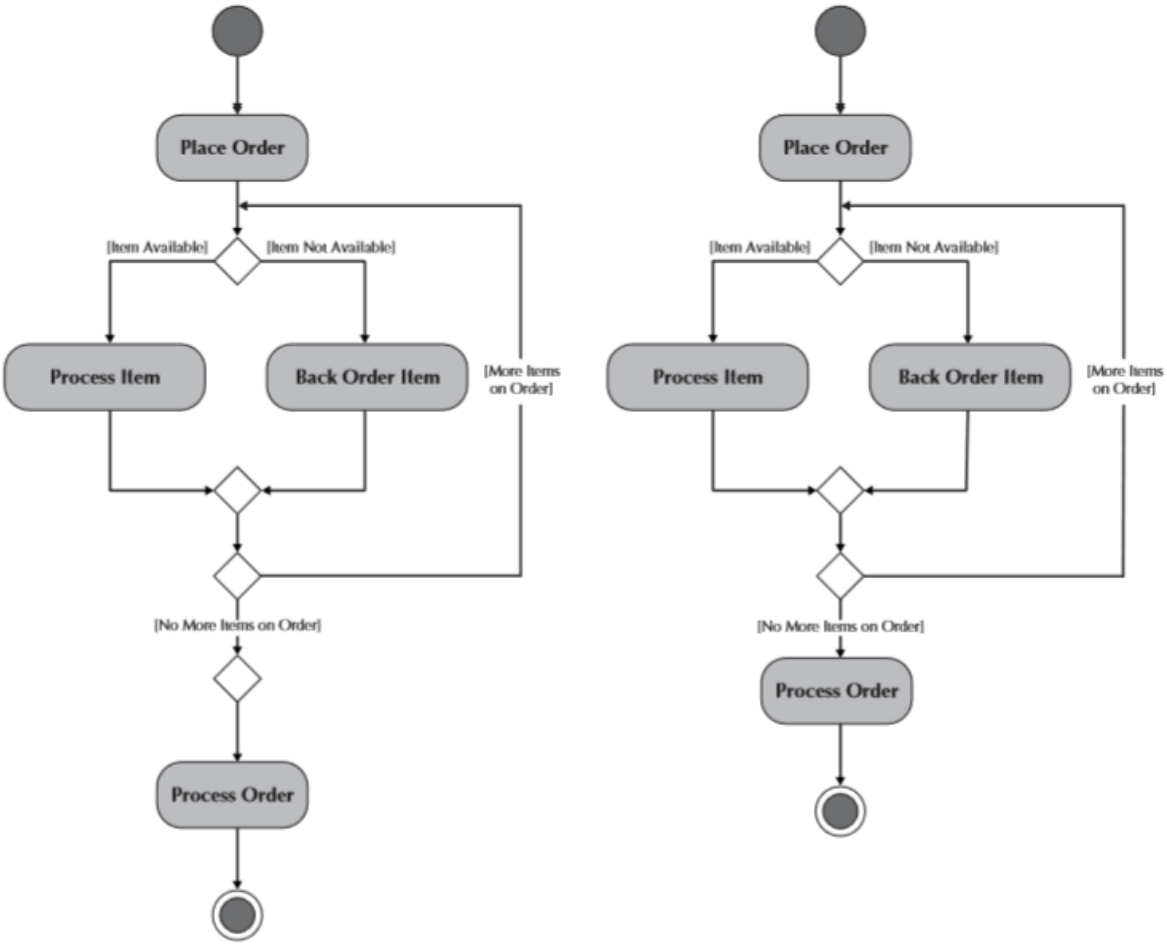
Aliran Kontrol dan Aliran Objek. Ada dua jenis aliran yang berbeda dalam diagram aktivitas: kontrol dan objek (lihat Gambar 4-7). Aliran kontrol memodelkan jalur eksekusi melalui proses bisnis. Aliran kontrol digambarkan sebagai garis padat dengan panah di atasnya yang menunjukkan arah aliran. Aliran kontrol hanya dapat dilampirkan pada tindakan atau aktivitas. Gambar 4-8 menggambarkan serangkaian aliran kontrol melalui sistem penunjukan kantor dokter. Aliran objek memodelkan aliran objek melalui proses bisnis. Karena aktivitas dan tindakan memodifikasi atau mengubah objek, aliran objek diperlukan untuk menunjukkan objek aktual yang mengalir masuk dan keluar dari tindakan atau aktivitas. Aliran objek digambarkan sebagai garis putus-putus dengan panah di atasnya yang menunjukkan arah aliran. Aliran objek individu harus dilampirkan ke tindakan atau aktivitas di satu ujung dan simpul objek di ujung lainnya. Gambar 4-8 menggambarkan satu set kontrol dan objek mengalir melalui sistem penunjukan kantor dokter.

Kontrol Node. Ada tujuh jenis node kontrol yang berbeda dalam diagram aktivitas: aktivitas awal, aktivitas akhir, aliran akhir, keputusan, penggabungan, percabangan, dan penggabungan (lihat Gambar 4-7). Sebuah node awal menggambarkan awal dari serangkaian

tindakan atau kegiatan. Node awal ditampilkan sebagai lingkaran kecil yang terisi. Node aktivitas akhir digunakan untuk menghentikan proses yang sedang dimodelkan. Setiap kali simpul aktivitas akhir tercapai, semua tindakan dan aktivitas segera diakhiri, terlepas dari apakah tindakan dan aktivitas tersebut telah selesai. Node aktivitas terakhir direpresentasikan sebagai lingkaran yang mengelilingi lingkaran kecil yang terisi, membuatnya menyerupai sasaran. Node aliran akhir mirip dengan node aktivitas akhir, kecuali bahwa node tersebut menghentikan jalur eksekusi tertentu melalui proses bisnis tetapi memungkinkan jalur paralel atau paralel lainnya untuk melanjutkan. Sebuah node aliran akhir ditampilkan sebagai lingkaran kecil dengan X di dalamnya.

Keputusan dan penggabungan node mendukung pemodelan struktur keputusan dari proses bisnis. Node keputusan digunakan untuk mewakili kondisi pengujian aktual yang menentukan jalur mana yang keluar dari simpul keputusan yang akan dilalui. Dalam hal ini, setiap jalur keluar harus diberi label dengan kondisi penjaga. Kondisi penjaga mewakili nilai pengujian untuk jalur tertentu yang akan dieksekusi. Misalnya, pada Gambar 4-8, simpul keputusan tepat di bawah aktivitas Dapatkan Informasi Pasien memiliki dua jalur eksklusif yang dapat dijalankan: satu untuk pasien lama, atau sebelumnya, dan yang lainnya untuk pasien baru. Node gabungan digunakan untuk menyatukan kembali beberapa jalur yang saling eksklusif yang telah dipisahkan berdasarkan keputusan sebelumnya (misalnya, jalur pasien lama dan baru pada Gambar 4-8 disatukan kembali di dekat bagian bawah diagram). Namun, terkadang, untuk kejelasan, lebih baik tidak menggunakan simpul gabungan. Sebagai contoh, pada Gambar 4-9, manakah dari dua diagram aktivitas, keduanya mewakili tingkat ikhtisar dari suatu proses pemesanan, yang lebih mudah dipahami, yang di sebelah kiri atau yang di sebelah kanan? Yang di sebelah kiri berisi simpul gabungan untuk pertanyaan Lebih Banyak Item tentang Pesanan, tetapi yang di sebelah kanan tidak. Dalam arti tertentu, simpul keputusan memainkan tugas ganda dalam diagram di sebelah kanan: Ini juga berfungsi sebagai simpul gabungan. Secara teknis, kita tidak boleh menghilangkan simpul gabungan; namun, terkadang secara teknis benar menurut aturan diagram UML sebenarnya menyebabkan diagram menjadi membingungkan. Dari perspektif pemodelan proses bisnis, banyak akal sehat bisa sangat membantu.

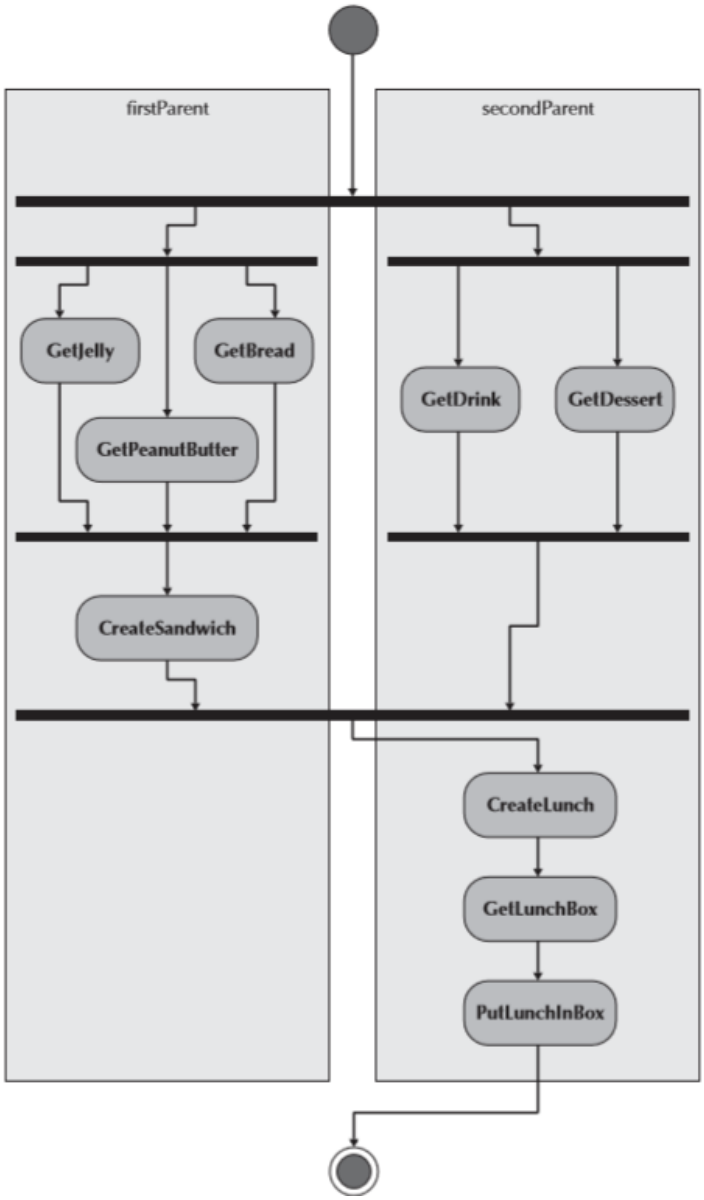
Fork dan join node memungkinkan proses paralel dan konkuren dimodelkan (lihat Gambar 4-7). Node garpu digunakan untuk membagi perilaku proses bisnis menjadi beberapa aliran paralel atau bersamaan. Tidak seperti simpul keputusan, jalur tidak saling eksklusif (yaitu, kedua jalur dieksekusi secara bersamaan). Misalnya, pada Gambar 4-10, simpul garpu digunakan untuk menunjukkan bahwa dua proses paralel yang bersamaan akan dieksekusi. Dalam hal ini, setiap proses dijalankan oleh dua prosesor terpisah (induk). Tujuan dari node bergabung mirip dengan node gabungan. Node bergabung hanya menyatukan kembali aliran paralel atau bersamaan yang terpisah dalam proses bisnis menjadi satu aliran.



Gambar 4-9 Dua Activity Diagram yang Sangat Mirip

Swimlanes. Diagram aktivitas dapat memodelkan proses bisnis yang independen dari implementasi objek apa pun. Namun, ada kalanya membantu untuk memecah diagram aktivitas sedemikian rupa sehingga dapat digunakan untuk memberikan tanggung jawab kepada objek atau individu yang benar-benar akan melakukan aktivitas tersebut. Ini sangat berguna saat memodelkan alur kerja bisnis dan dicapai melalui penggunaan swimlanes. Pada Gambar 4-10, swimlanes digunakan untuk memecah antara dua orang tua yang membuat makan siang sekolah yang terdiri dari sandwich selai kacang dan jelly, minuman, dan makanan penutup. Dalam hal ini, kami menggunakan swimlanes vertikal. Kita juga dapat menggambar diagram aktivitas menggunakan lebih banyak orientasi kiri-ke-kanan daripada orientasi atas-bawah. Dalam hal ini, swimlanes digambar secara horizontal.

Dalam alur kerja bisnis yang sebenarnya, akan ada aktivitas yang harus dikaitkan dengan peran individu yang terlibat dalam alur kerja bisnis (misalnya, karyawan atau pelanggan) dan aktivitas yang harus diselesaikan oleh sistem informasi yang dibuat. Asosiasi aktivitas dengan peran eksternal, peran internal, dan sistem ini sangat berguna saat membuat deskripsi Use-Case yang dijelaskan nanti dalam bab ini.



Gambar 4-10 Activity Diagram Membuat Kotak Makan Siang Sekolah

Pedoman Membuat Diagram Aktivitas

Scott Ambler menyarankan panduan berikut saat membuat diagram aktivitas:

- Karena diagram aktivitas dapat digunakan untuk memodelkan jenis proses apa pun, Anda harus mengatur konteks atau cakupan aktivitas yang dimodelkan. Setelah Anda menentukan ruang lingkup, Anda harus memberi diagram judul yang sesuai.
- Anda harus mengidentifikasi aktivitas, aliran kontrol, dan aliran objek yang terjadi di antara aktivitas.
- Anda harus mengidentifikasi setiap keputusan yang merupakan bagian dari proses yang dimodelkan.
- Anda harus mencoba mengidentifikasi prospek paralelisme dalam prosesnya.
- Anda harus menggambar diagram aktivitas.

Saat menggambar diagram aktivitas, diagram harus dibatasi pada satu simpul awal yang memulai proses yang dimodelkan. Node ini harus ditempatkan di kiri atas atau atas diagram, tergantung pada kompleksitas diagram. Untuk sebagian besar proses bisnis,

seharusnya hanya ada satu simpul aktivitas akhir. Node ini harus ditempatkan di bagian bawah atau kanan bawah diagram (lihat Gambar 4-8, 4-9, dan 4-10). Karena sebagian besar proses bisnis tingkat tinggi berurutan, tidak paralel, penggunaan simpul aliran akhir harus dibatasi.

Saat memodelkan proses bisnis atau alur kerja tingkat tinggi, hanya keputusan yang lebih penting yang harus dimasukkan dalam diagram aktivitas. Dalam kasus tersebut, kondisi penjaga yang terkait dengan arus keluar dari simpul keputusan harus saling eksklusif. Arus keluar dan kondisi penjaga harus membentuk satu set lengkap (yaitu, semua nilai potensial dari keputusan terkait dengan salah satu arus).

Seperti dalam pemodelan keputusan, fork dan join harus dimasukkan hanya untuk mewakili aktivitas paralel yang lebih penting dalam proses. Misalnya, versi alternatif dari Gambar 4-10 mungkin tidak menyertakan garpu dan gabungan yang terkait dengan aktivitas Get Jelly, Get Bread, Get Peanut Butter, Get Drink, dan Get Dessert. Ini akan sangat menyederhanakan diagram.

Saat menyusun diagram aktivitas, persilangan garis harus diminimalkan untuk meningkatkan keterbacaan diagram. Kegiatan pada diagram juga harus ditata dalam urutan kiri ke kanan dan/atau atas ke bawah berdasarkan urutan pelaksanaan kegiatan. Misalnya, pada Gambar 4-10, aktivitas Create Sandwich berlangsung sebelum aktivitas Create Lunch.

Akhirnya, setiap aktivitas yang tidak memiliki arus keluar atau arus masuk harus ditantang. Kegiatan tanpa arus keluar disebut sebagai kegiatan lubang hitam. Jika aktivitas benar-benar merupakan titik akhir dalam diagram, aktivitas tersebut harus memiliki aliran kontrol darinya ke aktivitas akhir atau simpul aliran akhir. Aktivitas yang tidak memiliki aliran masuk dikenal sebagai aktivitas keajaiban. Dalam hal ini, aktivitas tersebut kehilangan aliran masuk baik dari simpul awal diagram atau dari aktivitas lain.

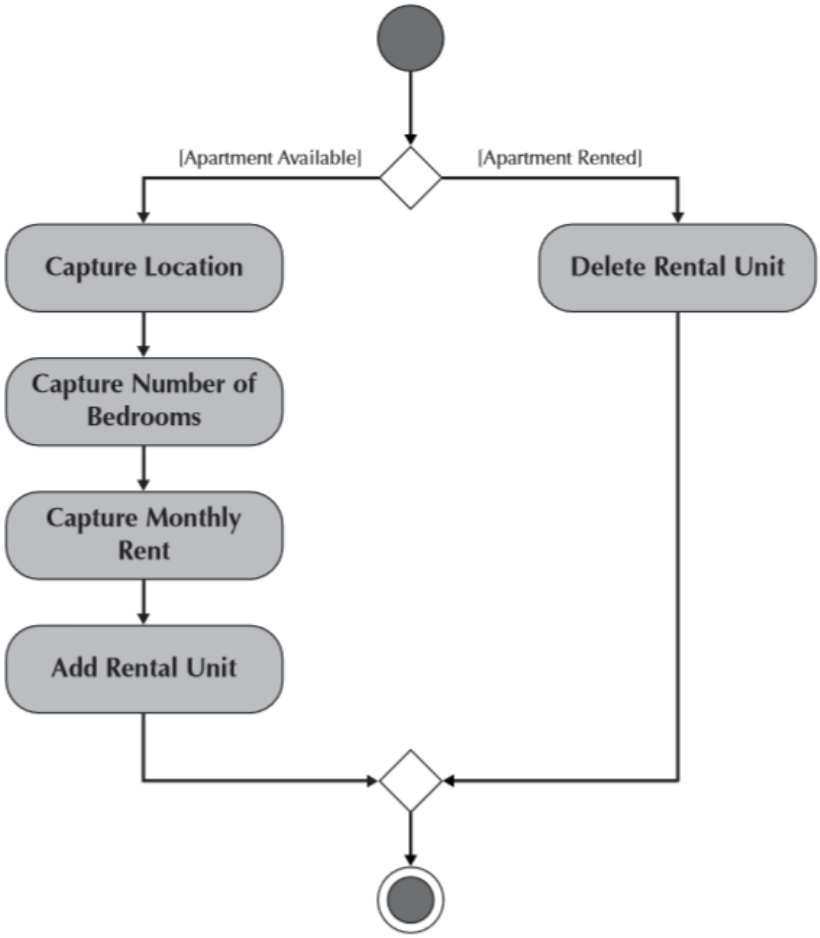
Membuat Diagram Aktivitas

Ada lima langkah dalam membuat diagram aktivitas untuk mendokumentasikan dan memodelkan proses bisnis. Pertama, Anda harus memilih proses bisnis yang sebelumnya diidentifikasi untuk dimodelkan. Untuk melakukan ini, Anda harus meninjau definisi persyaratan (lihat Gambar 3-1) dan diagram Use-Case (lihat Gambar 4-2, 4-3, dan 4-4) yang dibuat untuk mewakili persyaratan. Anda juga harus meninjau semua dokumentasi yang dibuat selama proses pengumpulan persyaratan (lihat Bab 3), misalnya, laporan yang dibuat yang mendokumentasikan wawancara atau pengamatan, keluaran apa pun dari sesi JAD, analisis kuesioner apa pun yang digunakan, dan kartu cerita atau daftar tugas dibuat. Dalam kebanyakan kasus, Use-Case pada diagram Use-Case akan menjadi tempat terbaik untuk memulai. Misalnya, dalam sistem janji temu, kami telah mengidentifikasi tiga Use-Case utama: Kelola Janji Temu, Jadwal Produksi, dan Rekam Ketersediaan Dokter. Kami juga mengidentifikasi seluruh rangkaian Use-Case minor (ini akan berguna dalam mengidentifikasi elemen diagram aktivitas).

Kedua, mengidentifikasi rangkaian aktivitas yang diperlukan untuk mendukung proses bisnis. Misalnya, pada Gambar 3-1, tiga proses diidentifikasi sebagai bagian dari proses bisnis Kelola Janji Temu. Juga, dengan meninjau diagram Use-Case (lihat Gambar 4-4), kita melihat bahwa lima Use-Case minor terkait dengan Use-Case utama Kelola Janji Temu. Berdasarkan informasi ini, kita dapat mengidentifikasi serangkaian kegiatan. Dalam hal ini kegiatan yang dilakukan adalah Update Informasi Pasien, Melakukan Pengaturan Pembayaran, Membuat Pasien Baru, Membuat Janji Temu, Membatalkan Janji Temu, dan Mengubah Janji Temu.

Ketiga, mengidentifikasi aliran kontrol dan node yang diperlukan untuk mendokumentasikan logika proses bisnis. Misalnya, pada Gambar 4-4, Use-Case Buat

Pengaturan Pembayaran dan Perbarui Informasi Pasien adalah ekstensi dari Use-Case Kelola Janji Temu dan Buat Aplikasi Pasien Lama. Kita tahu bahwa Use-Case ini dijalankan hanya dalam keadaan tertentu. Dari sini kita dapat menyimpulkan bahwa diagram aktivitas harus menyertakan beberapa keputusan dan menggabungkan node. Berdasarkan definisi persyaratan (lihat Gambar 3-1), kita dapat menyimpulkan kumpulan keputusan lain dan menggabungkan node berdasarkan aktivitas Buat Janji Temu, Batalkan Janji Temu, dan Ubah Janji Temu yang diidentifikasi pada langkah sebelumnya.



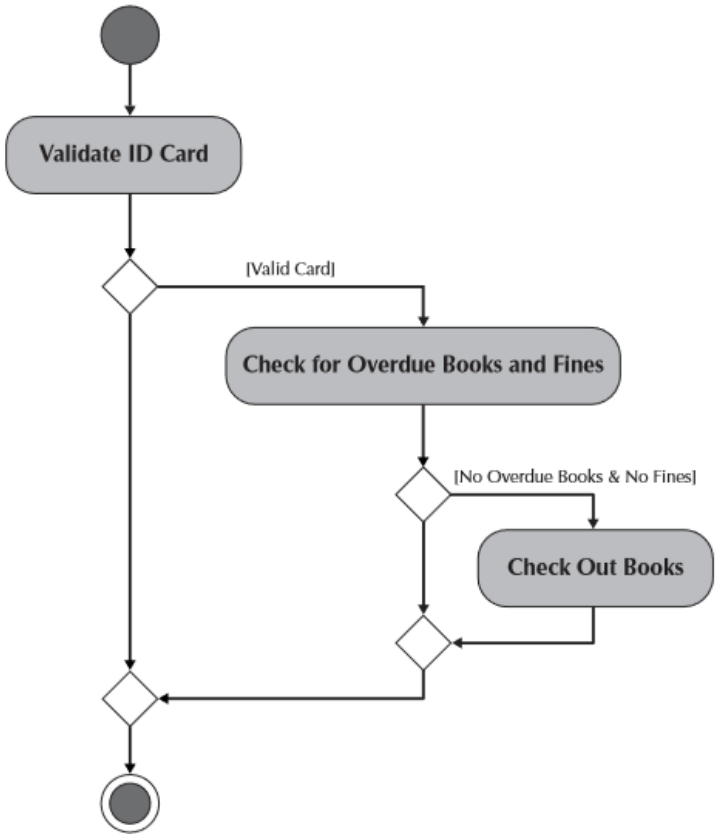
Gambar 4-11 Perumahan Kampus Menjaga Ketersediaan Unit Sewa Information Activity Diagram

Keempat, mengidentifikasi aliran objek dan node yang diperlukan untuk mendukung logika proses bisnis. Biasanya node dan aliran objek tidak ditampilkan pada banyak diagram aktivitas yang digunakan untuk memodelkan proses bisnis. Pengecualian utama adalah jika informasi yang ditangkap oleh sistem dalam satu aktivitas digunakan dalam aktivitas yang dilakukan kemudian, tetapi tidak segera setelah aktivitas yang menangkap informasi tersebut. Dalam contoh janji temu, jelas bahwa kita harus dapat menentukan apakah pasien adalah pasien lama atau baru dan jenis tindakan yang ingin dilakukan pasien (membuat, membatalkan, atau mengubah janji temu). Jelas bahwa pasien baru tidak dapat membatalkan atau mengubah janji karena pasien tersebut adalah pasien baru. Jelas, kita perlu menangkap jenis informasi ini di awal proses bisnis dan menggunakannya saat diperlukan. Misalnya, dalam masalah janji temu, kita perlu memiliki aktivitas Dapatkan Informasi Pasien yang menangkap informasi yang sesuai dan membuatnya tersedia pada waktu yang tepat dalam prosesnya.

Kelima, layout dan gambar diagram aktivitas untuk mendokumentasikan proses bisnis. Untuk alasan estetika dan pemahaman, seperti saat menggambar diagram Use-Case, Anda

harus berusaha meminimalkan potensi persilangan garis. Berdasarkan langkah-langkah sebelumnya dan dengan hati-hati meletakkan diagram, diagram aktivitas pada Gambar 4-8 dibuat untuk mendokumentasikan proses bisnis Kelola Janji Temu.

Contoh Perumahan Kampus. Langkah pertama dalam merinci proses bisnis yang diidentifikasi (Pertahankan Informasi Unit Sewa yang Tersedia dan Cari Unit Sewa yang Tersedia) adalah memilih salah satunya. Dalam contoh ini, kita akan fokus pada Menjaga Informasi Unit Sewa yang Tersedia yang terkait dengan pemilik apartemen. Berdasarkan uraian sebelumnya, terdapat dua aktivitas (subproses) terpisah: satu untuk menambah unit sewa dan satu untuk menghapus unit sewa. Untuk menambah unit sewa, pemilik apartemen harus menyediakan layanan perumahan kampus dengan lokasi apartemen, jumlah kamar tidur di apartemen, dan sewa bulanan apartemen. Untuk menghapus apartemen, pemilik apartemen harus memberi tahu layanan perumahan kampus bahwa apartemen tertentu telah disewa dan tidak lagi tersedia. Dengan menggunakan informasi ini, diagram aktivitas yang mewakili deskripsi logis dari Use-Case Informasi Pemeliharaan Unit Sewa yang Tersedia digambarkan pada Gambar 4-11. Perhatikan bahwa sama sekali tidak ada referensi untuk mengisi formulir, memasukkan informasi ke dalam database, atau mencari database. Sebenarnya ada banyak cara potensial yang berbeda di mana informasi apartemen dapat ditangkap, misalnya, pada formulir manual, pada formulir terkomputerisasi, pada formulir Web, dan melalui antarmuka seluler. Bahkan, kita mungkin ingin dapat mendukung mereka semua. Juga, ada banyak cara berbeda di mana informasi dapat disimpan. Namun, pada tahap pengembangan ini, semua detail desain dan implementasi harus diabaikan. Kami hanya tertarik untuk menangkap persyaratan fungsional. Setelah kami berhasil memodelkan persyaratan fungsional, kami dapat beralih ke persyaratan nonfungsional, desain, dan detail implementasi. Kami akan kembali ke contoh ini di bagian berikutnya dari bab ini. Namun, sebelum kita melanjutkan, selanjutnya kami menjelaskan diagram aktivitas untuk Use-Case Pinjam Buku dari masalah perpustakaan universitas.



Gambar 4-12 Activity Diagram Use Case Pinjam Buku

Contoh Perpustakaan. Seperti contoh Perumahan Kampus, langkah pertama adalah memilih proses bisnis yang akan dimodelkan. Dalam hal ini, kami ingin membuat diagram aktivitas untuk use case Pinjam Buku (lihat Gambar 4-6). Persyaratan fungsional untuk use case ini adalah:

> Kegiatan peminjaman dibangun di sekitar memeriksa buku dan mengembalikan buku oleh peminjam. Ada tiga jenis peminjam: mahasiswa, dosen atau staf, dan tamu. Terlepas dari jenis peminjam, peminjam harus memiliki KTP yang masih berlaku. Jika peminjam adalah mahasiswa, sistem akan memeriksa dengan database mahasiswa pendaftar memvalidasi kartu ID. Jika peminjam adalah staf pengajar atau staf, sistem memeriksa dengan database karyawan kantor personalia memvalidasi kartu identitas. Jika peminjam adalah tamu, kartu ID diperiksa terhadap database peminjam perpustakaan itu sendiri. Jika KTP masih berlaku, sistem juga harus memeriksa untuk menentukan apakah peminjam memiliki buku yang terlambat atau denda yang belum dibayar. Jika KTP tidak berlaku, peminjam memiliki buku yang jatuh tempo, atau peminjam memiliki denda yang belum dibayar, sistem harus menolak permintaan peminjam untuk memeriksa buku, jika tidak, permintaan peminjam harus dipenuhi.

Langkah kedua untuk memodelkan proses bisnis adalah mengidentifikasi aktivitas yang membentuk proses tersebut. Berdasarkan persyaratan untuk use case Pinjam Buku, kita dapat mengidentifikasi tiga kegiatan utama: Validasi KTP, Cek Buku dan Denda yang Jatuh Tempo, dan Check Out Buku. Langkah ketiga adalah mengidentifikasi aliran kontrol dan node kontrol yang diperlukan untuk memodelkan logika keputusan proses bisnis. Dalam hal ini, jelas harus ada simpul awal, simpul aliran akhir, dan satu set simpul keputusan dan gabungan untuk setiap keputusan yang akan dibuat. Langkah keempat adalah mengidentifikasi aliran objek dan node objek yang diperlukan untuk melengkapi deskripsi proses bisnis. Dalam hal ini, sebenarnya tidak perlu menyertakan node dan aliran objek. Akhirnya, kita dapat menyusun diagram (lihat Gambar 4-12).

4.5 DOKUMENTASI PROSES BISNIS DENGAN USE-CASE DAN DESKRIPSI USE-CASE

Diagram Use-Case memberikan pandangan menyeluruh tentang fungsionalitas dasar dari proses bisnis yang terkandung dalam sistem yang berkembang. Diagram aktivitas, dalam arti tertentu, membuka kotak hitam dari setiap proses bisnis dengan memberikan tampilan grafis yang lebih rinci dari aktivitas yang mendasari yang mendukung setiap proses bisnis. Deskripsi use-case menyediakan sarana untuk lebih lengkap mendokumentasikan aspek yang berbeda dari setiap use case individu. Deskripsi use-case didasarkan pada persyaratan yang diidentifikasi, diagram use-case, dan deskripsi diagram aktivitas dari proses bisnis. Deskripsi Use-Case berisi semua informasi yang diperlukan untuk mendokumentasikan fungsionalitas proses bisnis.

Use case adalah driver utama untuk semua teknik diagram UML. Sebuah use case mengkomunikasikan pada tingkat tinggi apa yang sistem perlu lakukan, dan semua teknik diagram UML membangun ini dengan menghadirkan fungsionalitas use-case dengan cara yang berbeda untuk tujuan yang berbeda. Use case adalah blok bangunan dimana sistem dirancang dan dibangun.

Use case menangkap interaksi khas sistem dengan pengguna sistem (pengguna akhir dan sistem lain). Interaksi ini mewakili pandangan eksternal, atau fungsional, dari sistem dari perspektif pengguna. Setiap use case menggambarkan satu dan hanya satu fungsi di mana pengguna berinteraksi dengan sistem. Meskipun use case mungkin berisi beberapa jalur yang dapat diambil pengguna saat berinteraksi dengan sistem, setiap jalur eksekusi yang mungkin melalui use case disebut sebagai skenario. Cara lain untuk melihat skenario adalah seolah-

olah skenario adalah contoh Use-Case tertentu. Skenario digunakan secara luas dalam pemodelan perilaku (lihat Bab 6). Terakhir, dengan mengidentifikasi semua skenario dan mencoba menjalankannya melalui kartu CRC permainan peran (lihat Bab 5), Anda akan menguji kejelasan dan kelengkapan pemahaman Anda yang berkembang tentang sistem yang sedang dikembangkan.

Saat membuat deskripsi Use-Case, tim proyek harus bekerja sama dengan pengguna untuk sepenuhnya mendokumentasikan persyaratan fungsional. Pengorganisasian persyaratan fungsional dan mendokumentasikannya dalam deskripsi use-case adalah proses yang relatif sederhana, tetapi membutuhkan banyak latihan untuk memastikan bahwa deskripsi cukup lengkap untuk digunakan dalam pemodelan struktural (Bab 5) dan perilaku (Bab 6). Tempat terbaik untuk memulai adalah meninjau Use-Case dan diagram aktivitas. Hal utama yang perlu diingat adalah bahwa setiap Use-Case dikaitkan dengan satu dan hanya satu peran yang dimiliki pengguna dalam sistem. Misalnya, resepsionis di kantor dokter mungkin memainkan banyak peran—dia dapat membuat janji, menjawab telepon, mengajukan catatan medis, menyambut pasien, dan sebagainya. Ada kemungkinan bahwa beberapa pengguna akan memainkan peran yang sama. Oleh karena itu, use case harus dikaitkan dengan peran yang dimainkan oleh pengguna dan bukan dengan pengguna itu sendiri.

Jenis Use-Case

Ada banyak jenis Use-Case yang berbeda. Kami menyarankan dua dimensi terpisah untuk mengklasifikasikan use case berdasarkan tujuan use case dan jumlah informasi yang terkandung dalam use case: gambaran umum versus detail dan esensial versus nyata.

Tinjauan Use-Case digunakan untuk memungkinkan analis dan pengguna menyetujui tinjauan persyaratan tingkat tinggi. Biasanya, ikhtisar use case dibuat sangat awal dalam proses memahami persyaratan sistem, dan mereka hanya mendokumentasikan informasi dasar tentang use case, seperti namanya; Nomor ID; aktor utama; Tipe; deskripsi singkat; dan hubungan antara aktor, aktor dan Use-Case, dan Use-Case. Ini dapat dengan mudah dibuat segera setelah pembuatan diagram use-case.

Setelah pengguna dan analis menyetujui ikhtisar persyaratan tingkat tinggi, ikhtisar Use-Case diubah menjadi Use-Case detail. Sebuah use case detail biasanya mendokumentasikan, sejauh mungkin, semua informasi yang dibutuhkan untuk use case. Ini dapat didasarkan pada aktivitas dan aliran kontrol yang terdapat dalam diagram aktivitas.

Use-Case esensial adalah kasus yang hanya menjelaskan masalah esensial minimum yang diperlukan untuk memahami fungsionalitas yang diperlukan. Sebuah Use-Case nyata berjalan lebih jauh dan menjelaskan serangkaian langkah-langkah tertentu. Misalnya, Use-Case penting di kantor dokter mungkin mengatakan bahwa resepsionis harus berusaha mencocokkan waktu janji temu yang diinginkan pasien dengan waktu yang tersedia, sedangkan Use-Case nyata mungkin mengatakan bahwa resepsionis harus mencari tanggal yang tersedia di kalender menggunakan Google Kalender untuk menentukan apakah waktu janji temu yang diminta tersedia. Perbedaan utama adalah bahwa Use-Case esensial adalah implementasi independen, sedangkan Use-Case nyata adalah deskripsi rinci tentang bagaimana menggunakan sistem setelah diimplementasikan. Dengan demikian, Use-Case nyata cenderung hanya digunakan dalam desain, implementasi, dan pengujian.

Elemen Deskripsi Use-Case

Uraian Use-Case berisi semua informasi yang diperlukan untuk membangun diagram struktural (Bab 5) dan perilaku (Bab 6) yang mengikutinya, tetapi deskripsi tersebut mengungkapkan informasi dengan cara yang kurang formal yang biasanya lebih mudah

dipahami pengguna. Gambar 4-13 menunjukkan contoh deskripsi Use-Case. Deskripsi use-case memiliki tiga bagian dasar: informasi ikhtisar, hubungan, dan aliran peristiwa.

Informasi Ikhtisar. Informasi ikhtisar mengidentifikasi use case dan memberikan informasi latar belakang dasar tentang use case. Nama Use-Case harus berupa frasa kata kerja-kata benda (mis., Make Old Patient Appt). Nomor ID Use-Case menyediakan cara unik untuk menemukan setiap Use-Case dan juga memungkinkan tim untuk melacak keputusan desain kembali ke persyaratan tertentu. Jenis use-case adalah gambaran umum atau detail dan esensial atau nyata. Aktor utama biasanya merupakan pemicu dari use case—orang atau hal yang memulai eksekusi dari use case. Tujuan utama dari use case adalah untuk memenuhi tujuan dari aktor utama. Deskripsi singkat biasanya satu kalimat yang menggambarkan esensi dari use case.

Tingkat kepentingan dapat digunakan untuk memprioritaskan Use-Case. Tingkat kepentingan memungkinkan pengguna untuk secara eksplisit memprioritaskan fungsi bisnis mana yang paling penting dan perlu menjadi bagian dari sistem versi pertama dan mana yang kurang penting dan dapat menunggu hingga versi yang lebih baru jika diperlukan. Tingkat kepentingan dapat menggunakan skala fuzzy, seperti tinggi, sedang, dan rendah (misalnya, pada Gambar 4-13 kami telah menetapkan tingkat kepentingan tinggi untuk Use-Case Make Old Patient Appt). Ini juga dapat dilakukan secara lebih formal dengan menggunakan rata-rata tertimbang dari serangkaian kriteria. Misalnya, Larman menyarankan untuk menilai setiap Use-Case berdasarkan kriteria berikut menggunakan skala dari nol hingga lima:

- Use case mewakili proses bisnis yang penting.
- Use case mendukung perolehan pendapatan atau pengurangan biaya.
- Teknologi yang dibutuhkan untuk mendukung use case adalah hal baru atau berisiko dan oleh karena itu memerlukan penelitian yang cukup besar.
- Fungsionalitas yang dijelaskan dalam use case adalah kompleks, berisiko, dan/atau kritis terhadap waktu. Bergantung pada kompleksitas Use-Case, mungkin berguna untuk mempertimbangkan untuk membagi implementasinya menjadi beberapa versi yang berbeda.
- Use-Case dapat meningkatkan pemahaman tentang desain yang berkembang relatif terhadap upaya yang dikeluarkan.

Sebuah use case mungkin memiliki banyak pemegang kepentingan yang memiliki kepentingan dalam use case tersebut. Setiap use case mencantumkan masing-masing pemegang kepentingan dengan minat masing-masing dalam use case (misalnya, Pasien Lama dan Dokter). Daftar pemegang kepentingan selalu menyertakan aktor utama (mis., Pasien Lama).

Setiap use case biasanya memiliki pemicu—peristiwa yang menyebabkan use case dimulai (misalnya, Pasien Lama menelepon dan meminta janji baru atau meminta untuk membatalkan atau mengubah janji yang sudah ada). Pemicu dapat berupa pemicu eksternal, seperti pelanggan yang melakukan pemesanan atau alarm kebakaran berdering, atau dapat berupa pemicu sementara, seperti buku yang jatuh tempo di perpustakaan atau kebutuhan untuk membayar sewa.

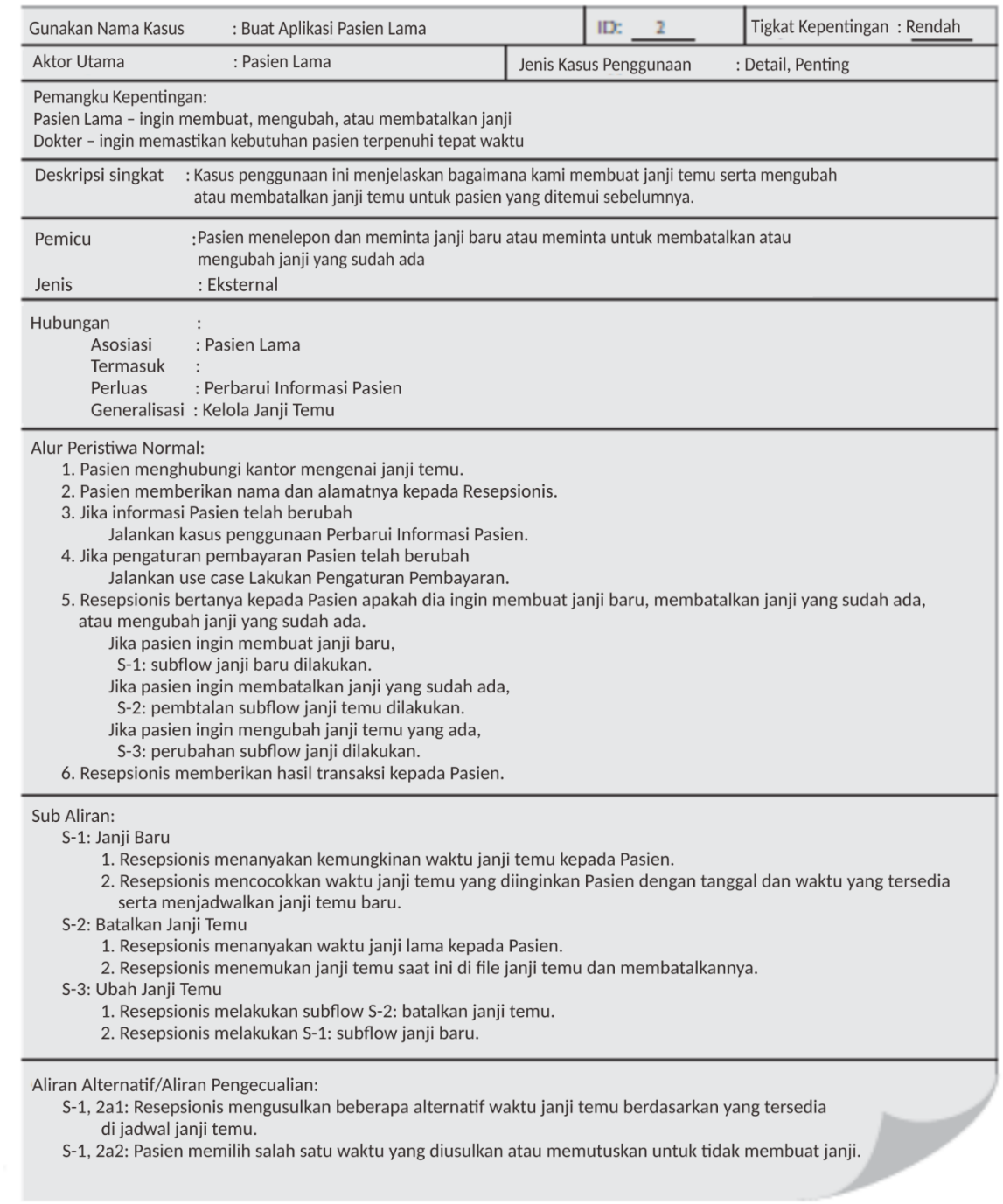
Hubungan. Hubungan use-case menjelaskan bagaimana use case terkait dengan use case dan pengguna lain. Ada empat tipe dasar hubungan: asosiasi, perluasan, penyertaan, dan generalisasi. Sebuah hubungan asosiasi mendokumentasikan komunikasi yang terjadi antara use case dan aktor yang menggunakan use case. Aktor adalah representasi UML untuk peran yang dimainkan pengguna dalam Use-Case. Misalnya, pada Gambar 4-13, use case Make Old Patient Appt dikaitkan dengan aktor Old Patient (lihat Gambar 4-4). Dalam hal ini, pasien

membuat janji. Semua aktor yang terlibat dalam use case didokumentasikan dengan hubungan asosiasi.

Hubungan include merepresentasikan inklusi wajib dari use case lain. Hubungan include memungkinkan dekomposisi fungsional—pemecahan use case yang kompleks menjadi beberapa use case yang lebih sederhana. Misalnya, pada Gambar 4-4, use case Kelola Jadwal dianggap kompleks dan cukup lengkap untuk diperhitungkan sebagai use case terpisah yang dapat dieksekusi oleh use case Produce Schedules dan Record Availability. Hubungan include juga memungkinkan bagian dari use case untuk digunakan kembali dengan membuatnya sebagai use case yang terpisah.

Perpanjangan hubungan mewakili perluasan fungsionalitas Use-Case untuk menggabungkan perilaku opsional. Pada Gambar 4-13, use case Make Old Patient Appt secara kondisional menggunakan use case Perbarui Informasi Pasien. Use case ini dijalankan hanya jika informasi pasien telah berubah.

Hubungan generalisasi memungkinkan Use-Case untuk mendukung pewarisan. Misalnya, Use-Case pada Gambar 4-4, Use-Case Kelola Janji Temu, dispesialisasikan sehingga pasien baru akan dikaitkan dengan Aplikasi Buat Pasien Baru dan pasien lama dapat dikaitkan dengan Aplikasi Buat Pasien Lama. Perilaku umum, atau umum, yang terkandung dalam Use-Case Buat Janji Pasien Baru dan Buat Janji Pasien Lama akan ditempatkan dalam Use-Case Kelola Janji Temu umum. Dengan kata lain, Use-Case Buat Janji Pasien Baru dan Buat Janji Pasien Lama akan mewarisi fungsionalitas umum dari Use-Case Kelola Janji Temu. Perilaku khusus akan ditempatkan dalam Use-Case khusus yang sesuai. Misalnya, perluasan hubungan ke Use-Case Perbarui Informasi Pasien akan ditempatkan dengan Use-Case Buat Janji Pasien Lama khusus.



Gunakan Nama Kasus	: Buat Aplikasi Pasien Lama	ID: 2	Tigkat Kepentingan : Rendah
Aktor Utama	: Pasien Lama	Jenis Kasus Penggunaan	: Detail, Penting

Pemangku Kepentingan:
Pasien Lama – ingin membuat, mengubah, atau membatalkan janji
Dokter – ingin memastikan kebutuhan pasien terpenuhi tepat waktu

Deskripsi singkat : Kasus penggunaan ini menjelaskan bagaimana kami membuat janji temu serta mengubah atau membatalkan janji temu untuk pasien yang ditemui sebelumnya.

Pemicu : Pasien menelepon dan meminta janji baru atau meminta untuk membatalkan atau mengubah janji yang sudah ada

Jenis : Eksternal

Hubungan :
- Asosiasi : Pasien Lama
- Termasuk :
- Perluas : Perbarui Informasi Pasien
- Generalisasi : Kelola Janji Temu

Alur Peristiwa Normal:
1. Pasien menghubungi kantor mengenai janji temu.
2. Pasien memberikan nama dan alamatnya kepada Resepsionis.
3. Jika informasi Pasien telah berubah
   Jalankan kasus penggunaan Perbarui Informasi Pasien.
4. Jika pengaturan pembayaran Pasien telah berubah
   Jalankan use case Lakukan Pengaturan Pembayaran.
5. Resepsionis bertanya kepada Pasien apakah dia ingin membuat janji baru, membatalkan janji yang sudah ada, atau mengubah janji yang sudah ada.
   Jika pasien ingin membuat janji baru,
   S-1: subflow janji baru dilakukan.
   Jika pasien ingin membatalkan janji yang sudah ada,
   S-2: pembtalan subflow janji temu dilakukan.
   Jika pasien ingin mengubah janji temu yang ada,
   S-3: perubahan subflow janji dilakukan.
6. Resepsionis memberikan hasil transaksi kepada Pasien.

Sub Aliran:
S-1: Janji Baru
1. Resepsionis menanyakan kemungkinan waktu janji temu kepada Pasien.
2. Resepsionis mencocokkan waktu janji temu yang diinginkan Pasien dengan tanggal dan waktu yang tersedia serta menjadwalkan janji temu baru.

S-2: Batalkan Janji Temu
1. Resepsionis menanyakan waktu janji lama kepada Pasien.
2. Resepsionis menemukan janji temu saat ini di file janji temu dan membatalkannya.

S-3: Ubah Janji Temu
1. Resepsionis melakukan subflow S-2: batalkan janji temu.
2. Resepsionis melakukan S-1: subflow janji baru.

Aliran Alternatif/Aliran Pengecualian:
S-1, 2a1: Resepsionis mengusulkan beberapa alternatif waktu janji temu berdasarkan yang tersedia di jadwal janji temu.
S-1, 2a2: Pasien memilih salah satu waktu yang diusulkan atau memutuskan untuk tidak membuat janji.

Gambar 4-13 Contoh Deskripsi Use-Case

Alur Peristiwa Akhirnya, langkah-langkah individu dalam proses bisnis dijelaskan. Tiga kategori yang berbeda dari langkah, atau aliran peristiwa, dapat didokumentasikan: aliran normal peristiwa, sub-aliran, dan alternatif, atau luar biasa, arus:

- Aliran normal peristiwa hanya mencakup langkah-langkah yang biasanya dieksekusi dalam Use-Case. Langkah-langkah yang tercantum dalam urutan di mana mereka dilakukan. Pada Gambar 4-13, pasien dan resepsionis melakukan percakapan mengenai nama pasien, alamat, dan tindakan yang akan dilakukan.
- Dalam beberapa kasus, aliran normal peristiwa harus didekomposisi menjadi satu set sub-aliran untuk menjaga aliran normal peristiwa sesederhana mungkin. Pada Gambar 4-13, kami telah mengidentifikasi tiga subflow: Buat Janji Temu, Batalkan Janji Temu, dan Ubah Janji Temu. Setiap langkah dari subflow terdaftar. Subflow ini didasarkan

pada logika aliran kontrol dalam representasi diagram aktivitas dari proses bisnis (lihat Gambar 4-7). Alternatifnya, kita dapat mengganti subflow dengan use case terpisah yang dapat digabungkan melalui hubungan include (lihat diskusi sebelumnya). Namun, ini harus dilakukan hanya jika use case yang baru dibuat masuk akal dengan sendirinya. Misalnya, pada Gambar 4-13, apakah masuk akal untuk memfaktorkan Use-Case Buat Janji Temu, Batalkan Janji Temu, dan/atau Ubah Janji Temu? Jika ya, maka subalur spesifik harus diganti dengan panggilan ke use case terkait, dan use case harus ditambahkan ke daftar hubungan include.

- Aliran alternatif atau luar biasa adalah aliran yang memang terjadi tetapi tidak dianggap sebagai norma. Ini harus didokumentasikan. Sebagai contoh, pada Gambar 4-13, kita telah mengidentifikasi dua aliran alternatif atau aliran luar biasa. Yang pertama hanya membahas situasi yang terjadi ketika rangkaian waktu janji temu yang diminta tidak tersedia. Yang kedua hanyalah langkah kedua ke aliran alternatif. Seperti halnya sub-aliran, tujuan utama pemisahan aliran alternatif atau aliran luar biasa adalah untuk menjaga aliran kejadian yang normal sesederhana mungkin. Sekali lagi, seperti halnya sub-alur, dimungkinkan untuk mengganti aliran alternatif atau luar biasa dengan Use-Case terpisah yang dapat diintegrasikan melalui hubungan perluasan (lihat diskusi sebelumnya).

Kapan peristiwa harus diperhitungkan dari aliran normal peristiwa menjadi sub-aliran? Kapan sub-aliran dan/atau aliran alternatif atau luar biasa diperhitungkan ke dalam Use-Case yang terpisah? Atau kapan segala sesuatunya dibiarkan begitu saja? Kriteria utama harus didasarkan pada tingkat kompleksitas yang dibutuhkan oleh use case. Semakin sulit untuk memahami use case, semakin besar kemungkinan kejadian harus diperhitungkan ke dalam subflow, atau subflow dan/atau aliran alternatif atau luar biasa harus diperhitungkan ke dalam use case terpisah yang disebut oleh use case saat ini. Ini, tentu saja, menciptakan lebih banyak Use-Case. Oleh karena itu, diagram use-case akan menjadi lebih berantakan. Dengan kata lain, pilihan yang harus diambil oleh analis adalah memiliki use case diagram yang lebih kompleks dengan use case yang lebih sederhana atau memiliki use case diagram yang lebih sederhana dengan use case yang lebih kompleks. Secara praktis, kita harus memutuskan mana yang lebih masuk akal. Ini sangat bervariasi, tergantung pada masalah dan klien. Ingat, kami mencoba untuk mewakili, dengan cara yang lengkap dan sesingkat mungkin, pemahaman kami tentang proses bisnis yang kami selidiki sehingga klien dapat memvalidasi persyaratan yang kami modelkan. Oleh karena itu, sebenarnya tidak ada satu jawaban yang benar. Itu sangat tergantung pada analis, klien, dan masalahnya.

Karakteristik Opsional. Karakteristik lain dari use case dapat didokumentasikan dengan deskripsi use case. Ini termasuk tingkat kerumitan use case; perkiraan jumlah waktu yang diperlukan untuk mengeksekusi use case; sistem dengan Use-Case yang terkait; aliran data spesifik antara aktor utama dan use case; atribut, batasan, atau operasi spesifik apa pun yang terkait dengan use case; setiap prasyarat yang harus dipenuhi agar use case dapat dieksekusi; atau jaminan apa pun yang dapat dibuat berdasarkan eksekusi use case. Seperti yang kami catat di awal bagian ini, tidak ada set standar karakteristik use case yang harus ditangkap. Kami menyarankan bahwa informasi yang terkandung dalam Gambar 4-13 adalah jumlah minimal yang harus ditangkap.

Panduan untuk Membuat Uraian Use-Case

Inti dari use case adalah aliran peristiwa. Menulis alur peristiwa dengan cara yang berguna untuk tahap perkembangan selanjutnya biasanya disertai dengan pengalaman.

Pertama, tulis setiap langkah individu dalam bentuk subjek-kata kerja-objek langsung dan, opsional, preposisi-objek tidak langsung. Bentuk ini kemudian dikenal sebagai kalimat

SVDPI. Bentuk kalimat ini terbukti berguna dalam mengidentifikasi kelas dan operasi (lihat Bab 5). Sebagai contoh, pada Gambar 4-13, langkah pertama dalam alur kejadian normal, Pasien menghubungi kantor mengenai janji temu, menyarankan kemungkinan tiga kelas objek: Pasien, Kantor, dan Janji Temu. Pendekatan ini menyederhanakan proses identifikasi kelas dalam model struktural (lihat Bab 5). Kalimat SVDPI tidak dapat digunakan untuk semua langkah, tetapi harus digunakan bila memungkinkan.

Kedua, memperjelas siapa atau apa yang memulai tindakan dan siapa atau apa penerima tindakan dalam setiap langkah. Biasanya, inisiator harus menjadi subjek kalimat dan penerima harus menjadi objek langsung dari kalimat. Misalnya, pada Gambar 4-13, langkah kedua, Pasien memberikan nama dan alamatnya kepada Resepsionis, dengan jelas menggambarkan Pasien sebagai inisiator dan Resepsionis sebagai penerima.

Ketiga, tulislah langkah tersebut dari sudut pandang pengamat independen. Untuk mencapai hal ini, setiap langkah mungkin harus ditulis terlebih dahulu dari sudut pandang inisiator dan penerima. Berdasarkan dua sudut pandang tersebut, kemudian dapat ditulis versi bird's-eye view. Misalnya, pada Gambar 4-13, Pasien memberikan nama dan alamatnya kepada Resepsionis, baik perspektif pasien maupun resepsionis tidak ditampilkan.

Keempat, tulis setiap langkah pada tingkat abstraksi yang sama. Setiap langkah harus membuat jumlah kemajuan yang sama dalam menyelesaikan Use-Case seperti setiap langkah lainnya. Pada Use-Case tingkat tinggi, jumlah kemajuan bisa sangat besar, sedangkan dalam Use-Case tingkat rendah, setiap langkah hanya dapat mewakili kemajuan tambahan. Misalnya, pada Gambar 4-13, setiap langkah mewakili jumlah usaha yang sama untuk diselesaikan.

Kelima, pastikan bahwa use case berisi serangkaian tindakan yang masuk akal. Setiap use case harus mewakili transaksi. Oleh karena itu, setiap use case harus terdiri dari empat bagian:

1. Aktor utama memulai eksekusi Use-Case dengan mengirimkan permintaan (dan mungkin data) ke sistem.
2. Sistem memastikan bahwa permintaan (dan data) valid.
3. Sistem memproses permintaan (dan data) dan mungkin mengubah keadaan internalnya sendiri.
4. Sistem mengirimkan aktor utama hasil pemrosesan.

Misalnya, pada Gambar 4-13, pasien meminta janji temu (langkah 1 dan 2), resepsionis menentukan apakah ada informasi pasien yang berubah atau tidak (langkah 3), resepsionis menentukan apakah pengaturan pembayaran pasien telah berubah atau tidak (langkah 4), resepsionis mengatur transaksi janji (langkah 5), dan resepsionis memberikan hasil transaksi kepada pasien (langkah 6).

Pedoman keenam adalah prinsip KISS. Jika use case menjadi terlalu kompleks dan/atau terlalu panjang, use case harus didekomposisi menjadi satu set use case. Selanjutnya, jika aliran normal dari peristiwa use case menjadi terlalu kompleks, subflow harus digunakan. Misalnya, pada Gambar 4-13, langkah kelima dalam aliran normal peristiwa cukup kompleks untuk menguraikannya menjadi tiga sub-aliran terpisah. Namun, perawatan harus diambil untuk menghindari kemungkinan membusuk terlalu banyak. Kebanyakan dekomposisi harus dilakukan dengan kelas (lihat Bab 5).

Pedoman ketujuh berkaitan dengan langkah-langkah yang berulang. Biasanya, dalam bahasa pemrograman, kami menempatkan definisi dan kontrol loop di awal loop. Namun, karena langkah-langkah usecase ditulis dalam bahasa Inggris yang sederhana, biasanya lebih baik menulis Ulangi langkah A sampai E sampai beberapa kondisi terpenuhi setelah langkah

E. Ini membuat use case lebih mudah dibaca oleh orang yang tidak terbiasa dengan pemrograman.

Membuat Deskripsi Use-Case

Use-Case memberikan pandangan menyeluruh tentang proses bisnis yang terkandung dalam sistem yang berkembang. Diagram use-case menggambarkan jalur komunikasi antara aktor dan sistem. Use case dan dokumentasi deskripsi use-case mereka cenderung digunakan untuk memodelkan konteks sistem dan persyaratan rinci untuk sistem. Meskipun tujuan utama use case adalah untuk mendokumentasikan persyaratan fungsional sistem, mereka juga digunakan sebagai dasar untuk menguji sistem yang berkembang. Di bagian ini, kami menyediakan serangkaian langkah yang dapat digunakan untuk memandu pembuatan deskripsi Use-Case yang sebenarnya untuk setiap Use-Case dalam diagram Use-Case berdasarkan definisi persyaratan dan diagram Use-Case dan aktivitas. Langkah-langkah ini dilakukan secara berurutan, tetapi tentu saja analis sering melakukan siklus di antara mereka secara iteratif saat dia berpindah dari satu use case ke use case lainnya.

Langkah pertama adalah memilih salah satu use case untuk didokumentasikan dengan deskripsi use case. Menggunakan tingkat kepentingan use case dapat membantu melakukan ini. Sebagai contoh, pada Gambar 4-13, use case Make Old Patient Appt memiliki tingkat kepentingan yang tinggi. Karena itu, ini harus menjadi salah satu Use-Case sebelumnya yang diperluas. Kriteria yang disarankan oleh Larman juga dapat digunakan untuk menetapkan prioritas Use-Case, seperti yang disebutkan sebelumnya. Pendekatan alternatif menyarankan bahwa setiap Use-Case harus dipilih oleh setiap anggota tim pengembangan. Dalam pendekatan ini, setiap anggota tim diberikan satu set "titik" yang dapat mereka gunakan untuk memilih Use-Case. Mereka dapat menggunakan semua titik mereka untuk memilih satu Use-Case, atau mereka dapat menyebarkannya ke serangkaian Use-Case. Use-Case kemudian dapat diurutkan berdasarkan jumlah titik yang diterima. Deskripsi Use-Case dibuat untuk Use-Case individu berdasarkan urutan peringkat.

Langkah kedua adalah membuat deskripsi gambaran umum dari use case; yaitu, beri nama aktor utama, atur jenis Use-Case, daftar semua pemegang kepentingan yang diidentifikasi dan kepentingan mereka dalam Use-Case, identifikasi tingkat kepentingan Use-Case, berikan deskripsi singkat tentang Use-Case, berikan informasi pemicu untuk Use-Case, dan daftar hubungan di mana Use-Case berpartisipasi.

Langkah ketiga adalah mengisi langkah-langkah aliran normal peristiwa yang diperlukan untuk menggambarkan setiap use case. Langkah-langkah tersebut berfokus pada apa yang dilakukan proses bisnis untuk menyelesaikan Use-Case, sebagai lawan dari tindakan apa yang dilakukan pengguna atau entitas eksternal lainnya. Secara umum, langkah-langkah harus dicantumkan dalam urutan pelaksanaannya, dari pertama hingga terakhir. Ingatlah untuk menulis langkah-langkah dalam formulir SVDPI bila memungkinkan. Dalam menulis use case, ingat tujuh pedoman yang dijelaskan sebelumnya. Tujuannya pada titik ini adalah untuk menggambarkan bagaimana use case yang dipilih beroperasi. Salah satu cara terbaik untuk mulai memahami bagaimana seorang aktor bekerja melalui use case adalah dengan memvisualisasikan melakukan langkah-langkah dalam use case—yaitu, role play. Teknik memvisualisasikan cara berinteraksi dengan sistem dan berpikir tentang cara kerja sistem lain (pembandingan informal) adalah teknik penting yang membantu analis dan pengguna memahami cara kerja sistem dan cara menulis Use-Case. Kedua teknik (visualisasi dan benchmarking informal) umum dalam praktik. Penting untuk diingat bahwa pada titik ini dalam pengembangan use case, kita hanya tertarik pada keberhasilan eksekusi use case yang tipikal. Jika kita mencoba memikirkan semua kemungkinan kombinasi kegiatan yang dapat berlangsung, kita tidak akan pernah mendapatkan apa pun yang tertulis. Pada titik ini, kami

hanya mencari tiga hingga tujuh langkah utama. Fokus hanya pada melakukan proses tipikal yang diwakili oleh use case.

Langkah keempat adalah memastikan bahwa langkah-langkah yang tercantum dalam alur kejadian normal tidak terlalu rumit atau terlalu panjang. Setiap langkah harus memiliki ukuran yang sama dengan yang lain. Misalnya, jika kita menulis langkah-langkah untuk menyiapkan makanan, langkah-langkah seperti mengambil garpu dari laci dan meletakkan garpu di atas meja jauh lebih kecil daripada menyiapkan kue menggunakan campuran. Jika kita berakhir dengan lebih dari tujuh langkah atau langkah yang ukurannya sangat bervariasi, kita harus kembali dan meninjau setiap langkah dengan hati-hati dan mungkin menulis ulang langkah-langkah tersebut.

Salah satu pendekatan yang baik untuk menghasilkan langkah-langkah untuk Use-Case adalah meminta pengguna memvisualisasikan diri mereka benar-benar melakukan Use-Case dan meminta mereka menuliskan langkah-langkah seolah-olah mereka sedang menulis resep untuk buku masak. Dalam kebanyakan kasus, pengguna akan dapat dengan cepat menentukan apa yang mereka lakukan dalam model apa adanya. Mendefinisikan langkah-langkah untuk Use-Case yang akan datang mungkin membutuhkan sedikit lebih banyak pelatihan. Dalam pengalaman kami, deskripsi langkah-langkahnya sangat berubah saat pengguna mengerjakan Use-Case. Saran kami adalah menggunakan papan tulis atau whiteboard (atau kertas dengan pensil) yang dapat dengan mudah dihapus untuk mengembangkan daftar langkah, dan kemudian menulis daftar pada formulir Use-Case. Itu harus ditulis pada formulir Use-Case hanya setelah serangkaian langkah didefinisikan dengan cukup baik.

Langkah kelima berfokus pada mengidentifikasi dan menulis aliran alternatif atau luar biasa. Aliran alternatif atau luar biasa adalah aliran kesuksesan yang mewakili perilaku opsional atau luar biasa. Mereka cenderung jarang terjadi atau sebagai akibat dari kegagalan aliran normal. Mereka harus diberi label sehingga tidak ada keraguan tentang aliran normal peristiwa mana yang terkait. Misalnya pada Gambar 4-13, aliran alternatif/luar biasa S-1, 2a1 dijalankan ketika langkah 2 dari sub-aliran S-1 gagal (yaitu, waktu janji temu yang diminta tidak tersedia). Seperti aliran normal dan aliran sub, aliran alternatif atau aliran luar biasa harus ditulis dalam bentuk SVDPI bila memungkinkan.

Langkah keenam adalah dengan hati-hati meninjau deskripsi use-case dan mengkonfirmasi bahwa use case sudah benar seperti yang tertulis, yang berarti meninjau use case dengan pengguna untuk memastikan setiap langkah sudah benar. Tinjauan harus mencari peluang untuk menyederhanakan use case dengan mendekomposisinya menjadi satu set use case yang lebih kecil, menggabungkannya dengan yang lain, mencari aspek umum baik dalam semantik dan sintaksis use case, dan mengidentifikasi use case baru. Ini juga saatnya untuk melihat ke dalam menambahkan hubungan include, extend, dan/atau generalisasi antara use case. Cara paling ampuh untuk mengonfirmasi use case adalah dengan meminta pengguna untuk bermain peran, atau menjalankan proses menggunakan langkah-langkah tertulis dalam use case. Analis menyerahkan potongan kertas pengguna yang diberi label dengan input utama ke use case dan meminta pengguna mengikuti langkah-langkah tertulis seperti resep untuk memastikan bahwa langkah-langkah tersebut benar-benar dapat menghasilkan output yang ditentukan untuk use case menggunakan inputnya.

Langkah ketujuh dan terakhir adalah mengulangi seluruh rangkaian langkah lagi. Pengguna sering berubah pikiran tentang apa itu use case dan apa yang termasuk di dalamnya. Sangat mudah untuk terjebak dalam detail pada saat ini, jadi ingatlah bahwa tujuannya hanya untuk menangani Use-Case utama. Oleh karena itu, analis harus terus mengulangi langkah-langkah ini sampai dia dan pengguna yakin bahwa jumlah Use-Case yang cukup telah

didokumentasikan untuk mulai mengidentifikasi kelas kandidat untuk model struktural (lihat Bab 5). Saat kelas kandidat diidentifikasi, kemungkinan Use-Case tambahan akan terungkap.

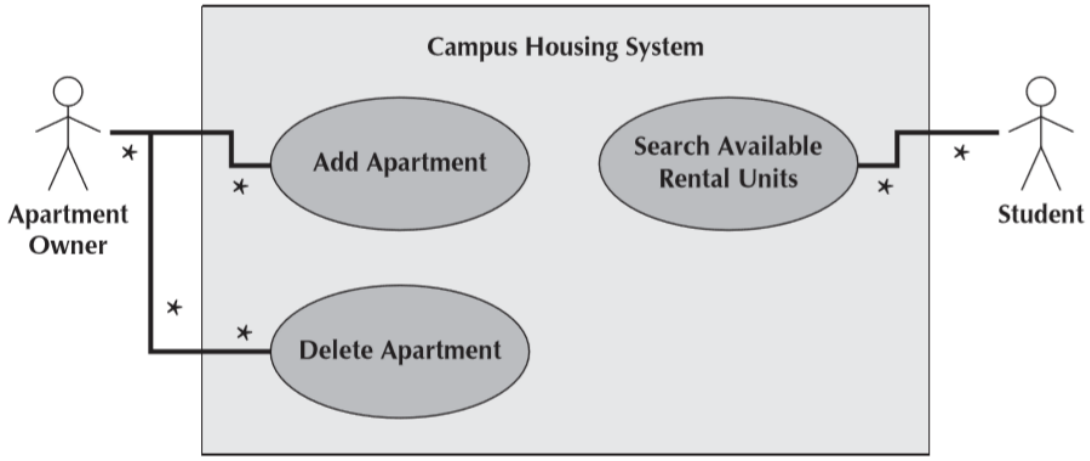
Contoh Perumahan Kampus Langkah pertama dalam mendokumentasikan use case dengan deskripsi use-case adalah memilih use case. Untuk tujuan instruksional, kami menggunakan Use-Case yang sama yang digunakan sebelumnya dengan diagram aktivitas; Menjaga Informasi Unit Sewa yang Tersedia, yang terkait dengan pemilik apartemen. Langkah selanjutnya adalah membuat Overview Description dari use case. Dalam hal ini, aktor utama adalah pemilik apartemen. Mengingat bahwa deskripsi use-case mendokumentasikan langkah-langkah logika rinci untuk use case, jenis use case adalah Detailed dan Essential. Pemegang kepentingan meliputi pemilik apartemen dan layanan perumahan kampus. Kepentingan mereka masing-masing adalah mengiklankan apartemen yang tersedia dan menyediakan layanan yang memungkinkan pemilik apartemen untuk menyewa apartemen yang tersedia. Deskripsi Singkat untuk Use-Case ini adalah "Use-Case ini menjelaskan bagaimana layanan perumahan kampus dapat mempertahankan daftar terbaru dari apartemen yang tersedia." Pemicu use case adalah ketika pemilik apartemen ingin menambah atau menghapus apartemen yang tersedia. Dengan demikian, pemicu "dipecat" dari luar sistem. Dalam hal ini, tindakan pemilik apartemen memicu Use-Case ini. Hanya ada satu hubungan Asosiasi antara use case ini dan Aktor Utamanya: Pemilik Apartemen. Gambar 4-14 mendokumentasikan informasi ini.

Selanjutnya, kami mendokumentasikan dan memverifikasi langkah-langkah logis yang diperlukan untuk berhasil menjalankan Use-Case ini. Artinya, kami mendokumentasikan aliran normal kejadian, memeriksa aliran normal kejadian (mungkin mengidentifikasi aliran sub), mengidentifikasi aliran alternatif atau pengecualian, dan kemudian dengan hati-hati meninjau deskripsi untuk memastikannya lengkap. Jika Anda ingat, dalam contoh khusus ini, pemilik apartemen memberikan informasi untuk menambahkan unit sewa ke unit sewa yang tersedia atau memberikan informasi yang mengidentifikasi unit yang tersedia yang tidak lagi tersedia dan perlu dihapus dari daftar unit sewa yang tersedia. Kedua proses ini diperlakukan sebagai dua subproses dari Use-Case Pertahankan Informasi Unit Sewa yang Tersedia. Sekarang kita harus menentukan mana dari subproses ini yang akan diperlakukan sebagai Aliran Normal Peristiwa dan mana yang diperlakukan sebagai Aliran Alternatif atau Luar Biasa. Namun, setelah refleksi lebih lanjut, pertanyaan apakah ini harus dipisahkan menjadi dua Use-Case independen atau apakah mereka harus tetap bersama harus diselidiki. Ini adalah contoh yang bagus di mana berpindah dari satu representasi (diagram aktivitas) ke representasi lain (deskripsi Use-Case) secara berulang dan bertahap menimbulkan masalah yang tidak mudah terlihat. Dalam contoh ini, mungkin lebih baik untuk mengganti Use-Case Pertahankan Informasi Unit Sewa yang Tersedia dengan dua Use-Case yang lebih sederhana: satu untuk menambahkan unit sewaan dan satu lagi untuk menghapus unit sewaan. Akibatnya, kita sekarang harus mengubah diagram use-case (lihat Gambar 4-15) dan membuat dua diagram aktivitas untuk menggantikan yang sebelumnya (lihat Gambar 4-16). Dan, kita harus membuat dua deskripsi usecase untuk menggantikan yang baru saja kita mulai (lihat Gambar 4-17 dan 4-18). Kami akan kembali ke contoh ini di bab berikutnya ketika kami mulai membuat model struktural untuk layanan perumahan kampus. Namun, selanjutnya kita kembali ke masalah perpustakaan universitas.

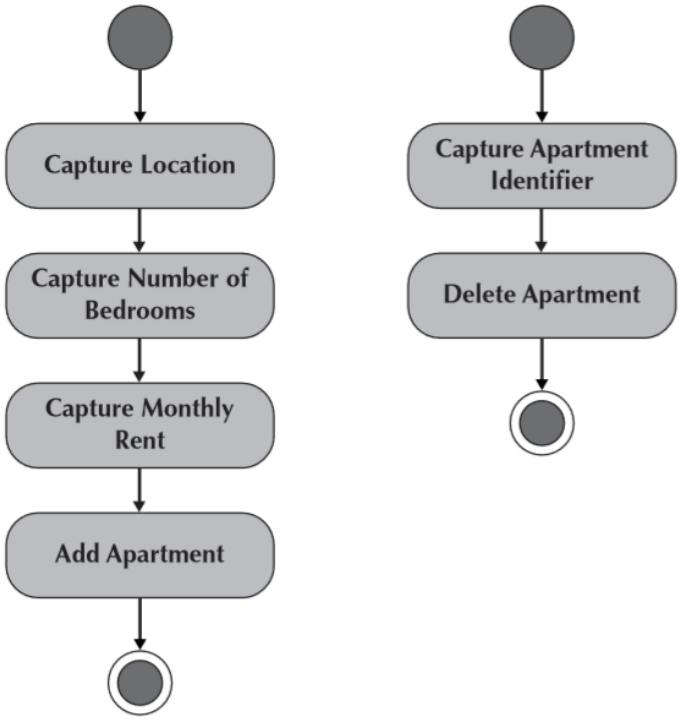
Nama Use Case : Pertahankan Informasi Unit Sewa yang Tersedia	ID : 1	Level Kesulitan : High
Primary Actor: Owner	Use-Case Type: Detail, Real	

Stakeholders and Interests:	Pemilik Apartemen—ingin mengiklankan apartemen yang tersedia Layanan Perumahan Kampus—menyediakan layanan yang memungkinkan pemilik apartemen untuk menyewa apartemen yang tersedia
Deskripsi Singkat:	Use-Case ini menjelaskan bagaimana layanan perumahan kampus dapat mempertahankan daftar terbaru dari apartemen yang tersedia.
Pemicu: Type :	Pemilik Apartemen ingin menambahkan apartemen yang tersedia External
Relationships: Generalization:	Association: Sales Rep Include: Extend:

Gambar 4-14 Perumahan Kampus Mempertahankan Informasi Unit Sewa yang Tersedia Gambaran Umum Deskripsi Use-Case



Gambar 4-15 Diagram Kasus Perumahan Kampus



Gambar 4-16 Campus Add and Delete Apartment Activity Diagram

Nama Use Case : Add Apartment	ID : 1	Level Kesulitan : High
Primary Actor: Owner	Use-Case Type: Detail, Real	
Stakeholders and Interests: Pemilik Apartemen—ingin mengiklankan apartemen yang tersedia Layanan Perumahan Kampus—menyediakan layanan yang memungkinkan pemilik apartemen untuk menyewa apartemen yang tersedia		
Deskripsi Singkat: Use-Case ini menjelaskan bagaimana layanan perumahan kampus dapat mempertahankan daftar terbaru dari apartemen yang tersedia.		
Pemicu: Pemilik Apartemen ingin menambahkan apartemen yang tersedia Type : External		
Relationships: Association: Sales Rep Include: Extend: Generalization:		
Alur Peristiwa Normal: 1. Menangkap lokasi apartemen. 2. Menangkap jumlah kamar tidur di apartemen.		

3. Menangkap sewa bulanan apartemen. 4. Tambahkan apartemen ke daftar apartemen yang tersedia.
Subflow:
Aliran Alternatif/Luar Biasa:

Gambar 4-17 Campus Housing Service Add an Apartment Use-Case Description

Nama Use Case : Delete Apartment	ID : 2	Level Kesulitan : High
Primary Actor: Owner	Use-Case Type: Detail, Real	
Stakeholders and Interests: Pemilik Apartemen—ingin menghapus apartemen Campus Housing Service—menyediakan layanan yang memungkinkan pemilik apartemen untuk menyewa apartemen yang tersedia		
Deskripsi Singkat: Use-Case ini menjelaskan bagaimana layanan perumahan kampus dapat mempertahankan daftar terbaru dari apartemen yang tersedia.		
Pemicu: Pemilik Apartemen ingin menghapus apartemen yang tersedia. Type : External		
Relationships: Association: Sales Rep Include: Extend: Generalization:		
Alur Peristiwa Normal: 5. Tangkap pengenal apartemen. 6. Hapus apartemen dari daftar apartemen yang tersedia		
Subflow:		
Aliran Alternatif/Luar Biasa:		

Gambar 4-18 Campus Housing Service Hapus Uraian Use-Case Apartemen

Contoh Perpustakaan. Seperti contoh Perumahan Kampus, langkah pertama untuk mendokumentasikan proses bisnis dengan deskripsi use-case adalah memilih use case. Karena kami sebelumnya memilih Use-Case Pinjam Buku dalam contoh Sistem Manajemen Koleksi Perpustakaan, kami akan tetap menggunakannya. Selanjutnya, kita perlu membuat deskripsi gambaran. Dalam hal ini, kita harus kembali dan melihat diagram use case (lihat Gambar 4-6)

yang menggambarkan perilaku eksternal Sistem Manajemen Koleksi Perpustakaan dan diagram aktivitas (lihat Gambar 4-12) yang menggambarkan fungsionalitas dari Use-Case Pinjam Buku. Ini juga merupakan ide yang baik untuk merujuk kembali, sekali lagi, ke persyaratan fungsional yang mendorong pembuatan use case Pinjam Buku. Di sini mereka:

Kegiatan peminjaman dibangun di sekitar memeriksa buku dan mengembalikan buku oleh peminjam. Ada tiga jenis peminjam: mahasiswa, dosen atau staf, dan tamu. Terlepas dari jenis peminjam, peminjam harus memiliki KTP yang masih berlaku. Jika peminjam adalah mahasiswa, sistem akan memeriksa dengan database mahasiswa pendaftar memvalidasi kartu ID. Jika peminjam adalah staf pengajar atau staf, sistem memeriksa dengan database karyawan kantor personalia memvalidasi kartu identitas. Jika peminjam adalah tamu, kartu ID diperiksa terhadap database peminjam perpustakaan itu sendiri. Jika KTP masih berlaku, sistem juga harus memeriksa untuk menentukan apakah peminjam memiliki buku yang terlambat atau denda yang belum dibayar. Jika KTP tidak berlaku, peminjam memiliki buku yang jatuh tempo, atau peminjam memiliki denda yang belum dibayar, sistem harus menolak permintaan peminjam untuk memeriksa buku, jika tidak, permintaan peminjam harus dipenuhi.

Berdasarkan tiga informasi penting ini dan menggunakan templat deskripsi Use-Case (lihat Gambar 4-13), kita dapat membuat deskripsi ikhtisar dari Use-Case Pinjam Buku (lihat Gambar 4-19).

Nama Use Case : Borrow Book	ID : 2	Level Kesulitan : High
Primary Actor: Borrower	Use-Case Type: Detail, Real	
Stakeholders and Interests: Peminjam—ingin memeriksa buku Pustakawan—ingin memastikan peminjam hanya mendapatkan buku yang layak		
Deskripsi Singkat: Use case ini menjelaskan bagaimana buku-buku dikeluarkan dari perpustakaan..		
Pemicu: Peminjam membawa buku untuk memeriksa meja.. Type : External		
Relationships: Association: Sales Rep Include: Extend: Generalization:		

Gambar 4-19 Deskripsi Ikhtisar Use Case Pinjam Buku

Dengan meninjau dengan cermat persyaratan fungsional (di atas) dan diagram aktivitas (Gambar 4-12), kita dapat dengan mudah mengidentifikasi Alur Peristiwa Normal untuk Use-Case Pinjam Buku. Selanjutnya, dimungkinkan untuk memutuskan apakah salah satu peristiwa yang terdapat dalam daftar Alur Normal Peristiwa harus didekomposisi menggunakan Subalur atau Use-Case lain yang perlu disertakan. Dalam kasus terakhir, kita

harus memodifikasi bagian Hubungan dari deskripsi ikhtisar dan memodifikasi diagram Use-Case untuk mencerminkan penambahan ini. Selain itu, berdasarkan struktur logika diagram aktivitas, dimungkinkan untuk mengidentifikasi aliran luar biasa alternatif ke aliran peristiwa normal untuk use case Pinjam Buku. Berdasarkan kesederhanaan keseluruhan dari use case Pinjam Buku, kami memutuskan untuk tidak menguraikan proses menggunakan subflows atau use case yang disertakan. Namun, karena struktur logika yang tercantum dalam diagram aktivitas, ada dua aliran alternatif/luar biasa yang diidentifikasi. Gambar 4-20 menggambarkan bagian Alur Normal dari Peristiwa, Subalur, dan Aliran Alternatif/Luar Biasa dari deskripsi Use-Case Buku Pinjam.

Alur Acara Normal: 1. Peminjam membawa buku ke Pustakawan di meja check out. 2. Peminjam memberikan Kartu Tanda Penduduk kepada Pustakawan. 3. Pustakawan memeriksa keabsahan KTP. Jika Peminjam adalah Peminjam Pelajar, Validasi KTP terhadap Database Panitera. Jika Peminjam adalah Peminjam Fakultas/Staf, Validasi KTP terhadap Database Kepegawaian. Jika Peminjam adalah Peminjam Tamu, Validasi KTP terhadap Database Tamu Perpustakaan. 4. Pustakawan memeriksa apakah Peminjam memiliki buku dan/atau denda yang jatuh tempo 5. Peminjam memeriksa buku
SubAliran:
Aliran Alternatif/Luar Biasa: 4a KTP tidak berlaku, permintaan buku ditolak. 5a Peminjam memiliki buku yang lewat jatuh tempo, denda, atau keduanya, permintaan pembukuan ditolak.

Gambar 4-20 Deskripsi Alur Use Case Pinjam Buku

4.6 VERIFIKASI DAN VALIDASI PROSES BISNIS DAN MODEL FUNGSIONAL

Sebelum kita beralih ke pemodelan struktural (Bab 5) dan perilaku (Bab 6), kita perlu memverifikasi dan memvalidasi rangkaian model fungsional saat ini untuk memastikan bahwa model tersebut mewakili proses bisnis yang sedang dipertimbangkan. Ini termasuk menguji kesetiaan setiap model; misalnya, kita harus yakin bahwa diagram aktivitas, deskripsi Use-Case, dan diagram Use-Case semuanya menggambarkan persyaratan fungsional yang sama. Sebelum kami menjelaskan tes khusus untuk dipertimbangkan, kami menjelaskan penelusuran, pendekatan manual yang mendukung verifikasi dan validasi model yang berkembang.

Verifikasi dan Validasi melalui Panduan

Panduan pada dasarnya adalah tinjauan sejawat terhadap suatu produk. Dalam kasus model fungsional, walkthrough adalah tinjauan dari berbagai model dan diagram yang dibuat selama pemodelan fungsional. Tinjauan ini biasanya diselesaikan oleh tim yang anggotanya berasal dari tim pengembangan dan klien. Tujuan dari walkthrough adalah untuk menguji secara menyeluruh kesetiaan model fungsional dengan persyaratan fungsional dan untuk memastikan bahwa model tersebut konsisten. Artinya, panduan mengungkap kesalahan atau kesalahan dalam spesifikasi yang berkembang. Namun, panduan tidak memperbaiki kesalahan—itu hanya mengidentifikasinya. Koreksi kesalahan harus diselesaikan oleh tim setelah penelusuran selesai.

Panduan sangat interaktif. Saat presenter berjalan melalui representasi, anggota tim walkthrough harus mengajukan pertanyaan tentang representasi. Misalnya, jika penyaji sedang menelusuri diagram aktivitas, anggota tim yang lain dapat bertanya mengapa aktivitas atau objek tertentu tidak disertakan. Proses sebenarnya dari hanya menyajikan representasi ke satu set mata baru dapat mengungkap kesalahpahaman dan kelalaian yang jelas. Dalam banyak kasus, pencipta representasi bisa tersesat di pohon pepatah dan tidak melihat hutan. Bahkan, seringkali tindakan berjalan melalui representasi menyebabkan presenter melihat sendiri kesalahannya. Untuk alasan psikologis, mendengar representasi membantu analis untuk melihat representasi lebih lengkap. Oleh karena itu, pencipta representasi harus secara teratur melakukan penelusuran model sendiri dengan membaca representasi keras-keras untuk diri mereka sendiri, terlepas dari bagaimana menurut mereka hal itu akan membuat mereka terlihat.

Ada peran tertentu yang dapat dimainkan oleh berbagai anggota tim panduan. Yang pertama adalah peran presenter. Ini harus dimainkan oleh orang yang terutama bertanggung jawab atas representasi spesifik yang sedang ditinjau. Individu ini mempresentasikan representasi ke tim panduan. Peran kedua adalah perekam, atau juru tulis. Perekam harus menjadi anggota tim analisis. Individu ini dengan hati-hati mencatat risalah rapat dengan mencatat semua peristiwa penting yang terjadi selama walkthrough. Secara khusus, semua kesalahan yang ditemukan harus didokumentasikan sehingga tim analisis dapat mengatasinya. Peran penting lainnya adalah memiliki seseorang yang mengangkat isu-isu mengenai pemeliharaan representasi. Yourdon mengacu pada individu ini sebagai oracle pemeliharaan. Karena penekanan pada reusability dalam pengembangan berorientasi objek, peran ini menjadi sangat penting. Akhirnya, seseorang harus bertanggung jawab untuk memanggil, mengatur, dan menjalankan rapat panduan.

Agar walkthrough berhasil, anggota tim walkthrough harus benar-benar siap. Semua bahan yang akan ditinjau harus didistribusikan dengan waktu yang cukup bagi anggota tim untuk meninjaunya sebelum pertemuan yang sebenarnya. Semua anggota tim diharapkan untuk menandai representasi sehingga selama walkthrough meeting, semua masalah yang relevan dapat didiskusikan. Jika tidak, walkthrough akan menjadi tidak efisien dan tidak efektif. Selama pertemuan yang sebenarnya, saat presenter berjalan melalui representasi, anggota tim harus menunjukkan kemungkinan kesalahan atau kesalahpahaman. Dalam banyak kasus, kesalahan dan kesalahpahaman disebabkan oleh asumsi yang tidak valid yang tidak akan terungkap tanpa panduan.

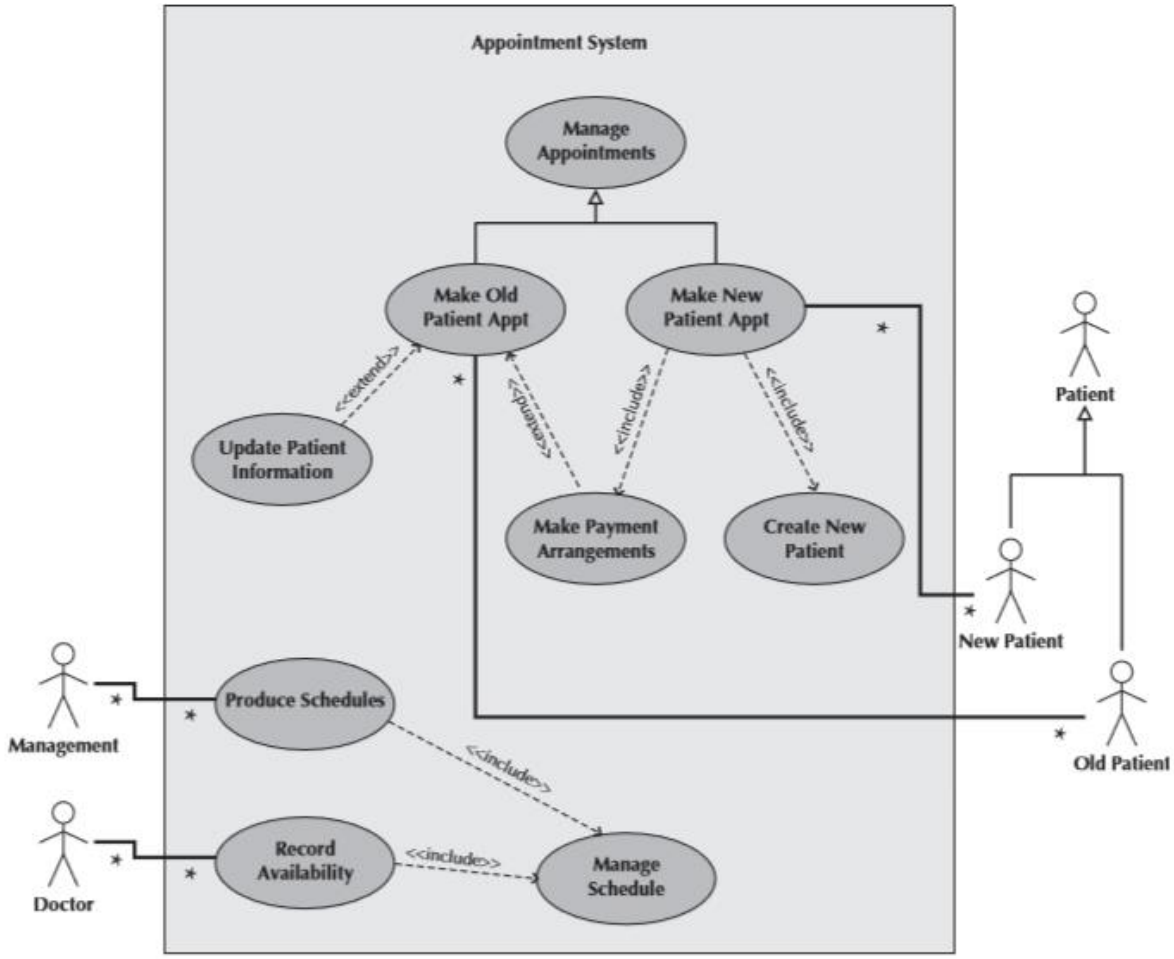
Salah satu potensi bahaya penelusuran adalah ketika manajemen memutuskan hasil dari pengungkapan kesalahan dalam representasi merupakan cerminan dari kemampuan analis. Ini harus dihindari dengan segala cara. Jika tidak, tujuan mendasar dari penelusuran—untuk meningkatkan kesetiaan representasi—akan digagalkan. Tergantung pada organisasinya, manajemen mungkin perlu dihilangkan dari proses penelusuran. Jika tidak, proses walkthrough bisa pecah menjadi slugfest untuk membuat beberapa anggota tim terlihat bagus dengan menghancurkan presenter. Untuk sedikitnya, ini jelas kontraproduktif.

Verifikasi dan Validasi Model Fungsional

Kami telah menyarankan tiga representasi berbeda untuk model fungsional: diagram aktivitas, deskripsi Use-Case, dan diagram Use-Case. Di bagian ini, kami menjelaskan seperangkat aturan untuk memastikan bahwa ketiga representasi ini konsisten di antara mereka sendiri.

Pertama, ketika membandingkan diagram aktivitas dengan deskripsi Use-Case, setidaknya harus ada satu peristiwa yang direkam dalam aliran normal peristiwa, sub-aliran, atau aliran alternatif/eksepsi dari deskripsi Use-Case untuk setiap aktivitas atau tindakan yang

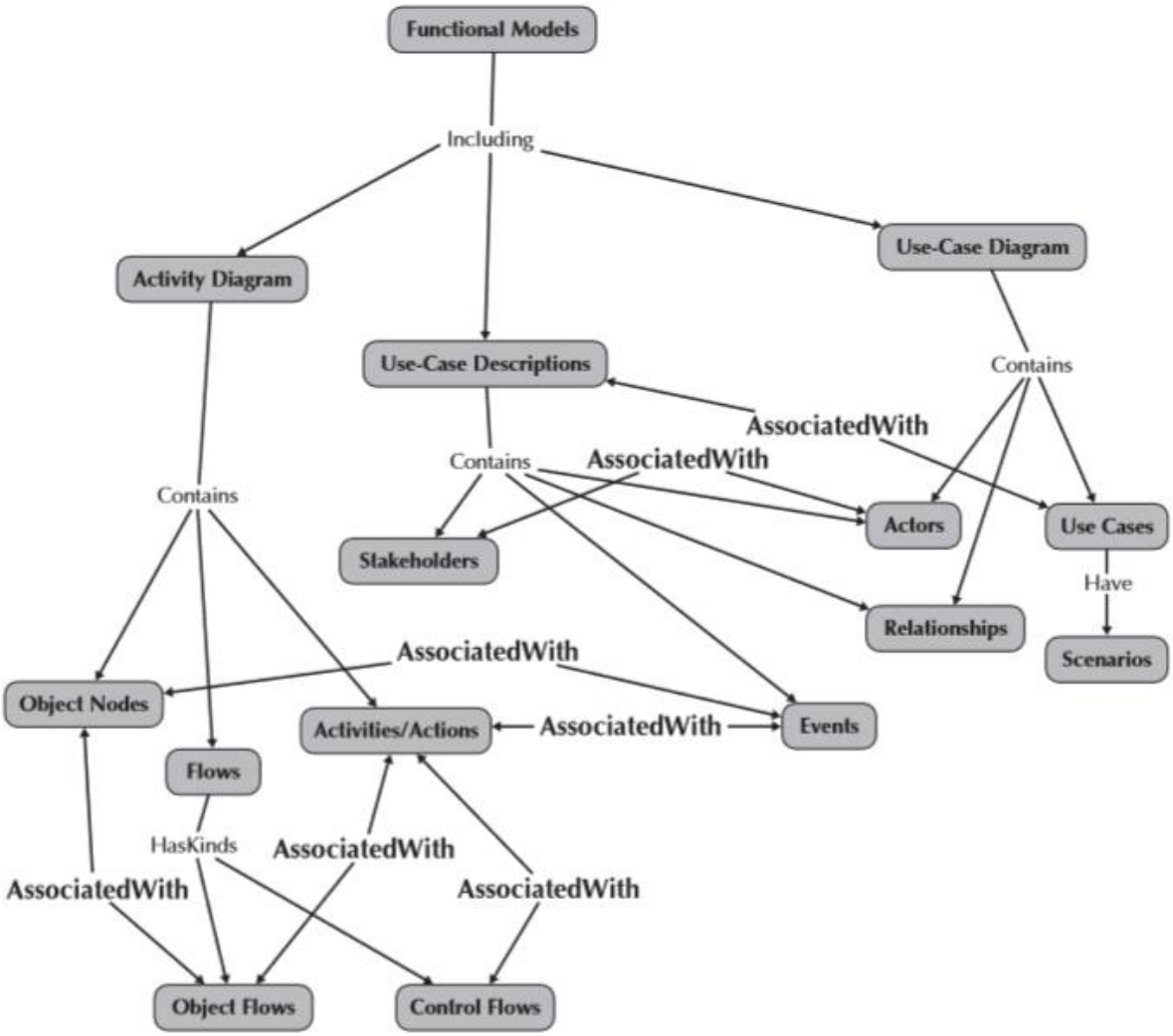
disertakan pada diagram aktivitas, dan setiap peristiwa harus dikaitkan dengan aktivitas atau tindakan. Misalnya, pada Gambar 4-4, ada aktivitas berlabel Dapatkan Informasi Pasien yang terkait dengan dua peristiwa pertama yang terdapat dalam aliran normal peristiwa dari deskripsi Use-Case yang ditunjukkan pada Gambar 4-13.



Gambar 4-21 Diagram Use-Case yang Dimodifikasi untuk Sistem Penunjukan

Kedua, semua objek yang digambarkan sebagai simpul objek dalam diagram aktivitas harus disebutkan dalam suatu peristiwa dalam aliran normal peristiwa, sub-aliran, atau aliran alternatif/luar biasa dari deskripsi use-case. Misalnya, diagram aktivitas pada Gambar 4-4 menggambarkan objek Appt, dan deskripsi usecase mengacu pada janji baru dan mengubah atau membatalkan janji yang sudah ada.

Ketiga, urutan kejadian yang berurutan dalam deskripsi use-case harus terjadi dalam urutan yang sama dari aktivitas yang terkandung dalam diagram aktivitas. Misalnya pada Gambar 4-4 dan 4-13, peristiwa yang terkait dengan aktivitas Dapatkan Informasi Pasien (peristiwa 1 dan 2) harus terjadi sebelum peristiwa yang terkait dengan aktivitas Lakukan Pengaturan Pembayaran (peristiwa 4).



Gambar 4-22 Hubungan Antar Model Fungsional

Keempat, ketika membandingkan deskripsi use-case dengan diagram use-case, harus ada satu dan hanya satu deskripsi use-case untuk setiap use case, dan sebaliknya. Misalnya, Gambar 4-13 menggambarkan deskripsi Use-Case dari Use-Case Make Old Patient Appt. Namun, diagram Use-Case yang ditunjukkan pada Gambar 4-4, diagram aktivitas yang ditunjukkan pada Gambar 4-8, dan deskripsi Use-Case yang diberikan pada Gambar 4-13 tidak konsisten satu sama lain. Dalam hal ini, diagram Use-Case menyiratkan bahwa Use-Case Lakukan Pengaturan Pembayaran adalah opsional terlepas dari apakah pasien tersebut adalah pasien baru atau lama. Namun, ketika kami meninjau diagram aktivitas, kami melihat bahwa itu adalah aktivitas opsional untuk pasien lama, tetapi aktivitas yang diperlukan untuk pasien baru. Oleh karena itu, hanya satu diagram yang benar. Dalam hal ini, diagram use-case perlu diperbaiki. Diagram Use-Case baru yang dikoreksi ditunjukkan pada Gambar 4-21.

Kelima, semua aktor yang tercantum dalam deskripsi use-case harus digambarkan pada diagram use-case. Setiap aktor harus memiliki link asosiasi yang menghubungkannya ke use case dan harus terdaftar dengan hubungan asosiasi dalam deskripsi use case. Misalnya, aktor Old Patient terdaftar dalam deskripsi use case dari use case Make Old Patient Appt (lihat Gambar 4-13), itu terdaftar dengan hubungan asosiasi dalam deskripsi usecase Make Old Patient Appt, dan itu terhubung ke use case dalam diagram use case (lihat Gambar 4-21).

Keenam, di beberapa organisasi, kita juga harus memasukkan pemegang kepentingan yang tercantum dalam deskripsi usecase sebagai aktor dalam diagram use-case. Misalnya,

mungkin ada hubungan antara aktor Doctor dan use case Make Old Patient Appt (lihat Gambar 4-13 dan 4-21). Namun, dalam kasus ini diputuskan untuk tidak menyertakan asosiasi ini karena Dokter tidak pernah berpartisipasi dalam use case Make Old Patient Appt.

Ketujuh, semua hubungan lain yang tercantum dalam deskripsi use-case (termasuk, perluasan, dan generalisasi) harus digambarkan pada diagram use-case. Misalnya, pada Gambar 4-13, ada hubungan perpanjangan yang terdaftar dengan use case Update Patient Information, dan pada Gambar 4-21, kita melihat bahwa itu muncul pada diagram antara dua use case.

Akhirnya, ada banyak persyaratan khusus diagram yang harus ditegakkan. Misalnya, dalam diagram aktivitas, simpul keputusan dapat dihubungkan ke simpul aktivitas atau tindakan hanya dengan aliran kontrol, dan untuk setiap simpul keputusan harus ada simpul gabungan yang cocok. Setiap jenis node dan aliran memiliki batasan yang berbeda. Namun, batasan lengkap untuk semua diagram UML berada di luar cakupan teks ini. Peta konsep pada Gambar 4-22 menggambarkan asosiasi di antara model fungsional.

Pertanyaan

1. Mengapa pemodelan proses bisnis itu penting?
2. Bagaimana Anda membuat Use-Case?
3. Mengapa Kita harus berusaha keras untuk memiliki sekitar tiga hingga sembilan Use-Case utama dalam proses bisnis?
4. Bagaimana Anda membuat diagram Use-Case?
5. Bagaimana diagram use-case terkait dengan pemodelan fungsional?
6. Jelaskan istilah-istilah berikut: aktor, use case, batas sistem, hubungan. Gunakan bahasa orang awam, seolah-olah Anda sedang menjelaskannya kepada pengguna.
7. Setiap asosiasi harus terhubung ke setidaknya satu _____ dan satu _____. Mengapa?
8. Apa saja heuristik untuk membuat diagram Use-Case?
9. Mengapa iterasi penting dalam membuat Use-Case?
10. Apa tujuan dari diagram aktivitas?
11. Apa perbedaan antara aktivitas dan tindakan?
12. Apa tujuan dari simpul garpu?
13. Apa saja jenis-jenis node kontrol?
14. Apa perbedaan antara aliran kontrol dan aliran objek?
15. Apa itu simpul objek?
16. Jelaskan bagaimana detail use case berbeda dari use case ikhtisar. Kapan masing-masing digunakan?
17. Bagaimana Use-Case esensial berbeda dari Use-Case nyata?
18. Apa elemen utama dari use case ikhtisar?
19. Apa elemen utama dari use case detail?
20. Apa sudut pandang dari use case, dan mengapa itu penting?
21. Apa sajakah pedoman untuk merancang satu set Use-Case? Berikan dua contoh perluasan asosiasi pada diagram use-case. Berikan dua contoh untuk asosiasi yang disertakan.
22. Manakah dari berikut ini yang bisa menjadi aktor yang ditemukan pada diagram use-case? Mengapa? Ms. Mary Smith Pemasok Pelanggan Pelanggan Internet Mr. John Seals Petugas entri data Administrator database
23. Apa itu CRUD? Mengapa berguna?
24. Apa itu walkthrough? Bagaimana kaitannya dengan verifikasi dan validasi?
25. Apa peran berbeda yang dimainkan selama penelusuran? Apa tujuan mereka?

26. Bagaimana model fungsional yang berbeda terkait, dan bagaimana hal ini mempengaruhi verifikasi dan validasi model?

Latihan

A. Selidiki situs web UML di Object Management Group (www.uml.org). Tulis ringkasan berita paragraf tentang status UML saat ini (mis., versi saat ini dan kapan akan dirilis, peningkatan di masa mendatang).
B. Selidiki Grup Manajemen Objek. Tulis memo singkat yang menjelaskan apa itu, tujuannya, dan pengaruhnya terhadap UML dan pendekatan objek untuk pengembangan sistem. (Petunjuk: Sumber yang bagus adalah www.omg.org.)
C. Gambarlah diagram Use-Case dan serangkaian diagram aktivitas untuk proses pembelian kacamata dari sudut pandang pasien. Langkah pertama adalah menemui dokter mata yang akan memberi Anda resep. Setelah Anda memiliki resep, Anda pergi ke apotek optik, di mana Anda memilih bingkai Anda dan memesan kacamata Anda. Setelah kacamata dibuat, Anda kembali ke toko untuk memasang dan membayar kacamata.
D. Buat satu set deskripsi Use-Case terperinci untuk proses pembelian kacamata di latihan C.
E. Gambarlah diagram Use-Case dan satu set diagram aktivitas untuk sistem kantor dokter berikut. Setiap kali pasien baru dilihat untuk pertama kalinya, mereka mengisi formulir informasi pasien yang menanyakan nama, alamat, nomor telepon, dan riwayat medis singkat mereka, yang disimpan dalam file informasi pasien. Ketika seorang pasien menelepon untuk menjadwalkan janji baru atau mengubah janji yang sudah ada, resepsionis memeriksa file janji untuk waktu yang tersedia. Setelah waktu yang tepat ditemukan untuk pasien, janji temu dijadwalkan. Jika pasien adalah pasien baru, entri yang tidak lengkap dibuat dalam file pasien; informasi lengkap akan dikumpulkan ketika pasien tiba untuk janji temu. Karena janji sering dibuat jauh sebelumnya, resepsionis biasanya mengirimkan kartu pos pengingat kepada setiap pasien dua minggu sebelum janji.
F. Buat satu set deskripsi Use-Case detail untuk sistem kantor dokter gigi dalam latihan E.
G. Gambarlah diagram Use-Case dan serangkaian diagram aktivitas untuk sistem pendaftaran universitas online. Sistem harus memungkinkan staf setiap departemen akademik untuk memeriksa mata kuliah yang ditawarkan oleh departemen mereka, menambah dan menghapus mata kuliah, dan mengubah informasi tentang mereka (misalnya, jumlah maksimum siswa yang diizinkan). Ini harus memungkinkan siswa untuk memeriksa kursus yang tersedia saat ini, menambah dan menghapus kursus ke dan dari jadwal mereka, dan memeriksa kursus tempat mereka terdaftar. Staf departemen harus dapat mencetak berbagai laporan tentang kursus dan siswa yang terdaftar di dalamnya. Sistem harus memastikan bahwa tidak ada siswa yang mengambil terlalu banyak mata kuliah dan bahwa siswa yang memiliki biaya yang belum dibayar tidak diizinkan untuk mendaftar (asumsikan bahwa data biaya dikelola oleh kantor keuangan universitas, yang diakses oleh sistem pendaftaran tetapi tidak berubah).
H. Buat satu set deskripsi Use-Case terperinci untuk sistem pendaftaran universitas online di latihan G.
I. Gambarlah diagram Use-Case dan serangkaian diagram aktivitas untuk sistem berikut. A Real Estate Inc. (AREI) menjual rumah. Orang yang ingin menjual rumahnya menandatangani kontrak dengan AREI dan memberikan informasi tentang rumahnya. Informasi ini disimpan dalam database oleh AREI, dan sebagian dari informasi ini

dikirim ke layanan daftar ganda seluruh kota yang digunakan oleh semua agen real estat. AREI bekerja dengan dua jenis pembeli potensial. Beberapa pembeli memiliki minat pada satu rumah tertentu. Dalam hal ini, AREI mencetak informasi dari databasenya, yang digunakan oleh agen real estate untuk membantu menunjukkan rumah kepada pembeli (suatu proses di luar cakupan sistem yang akan dimodelkan). Pembeli lain meminta saran AREI untuk menemukan rumah yang sesuai dengan kebutuhan mereka. Dalam hal ini, pembeli melengkapi formulir informasi pembeli yang dimasukkan ke dalam database pembeli, dan agen real estate AREI menggunakan informasinya untuk mencari database AREI dan layanan multi listing untuk rumah yang memenuhi kebutuhan mereka. Hasil pencarian tersebut dicetak dan digunakan untuk membantu agen real estate menunjukkan rumah kepada pembeli.

J. Buat satu set deskripsi Use-Case terperinci untuk sistem real estat dalam latihan I.

K. Lakukan langkah-langkah verifikasi dan validasi model fungsional sistem real estat yang dijelaskan dalam latihan I dan J.

L. Gambarlah diagram Use-Case dan serangkaian diagram aktivitas untuk sistem berikut. A Video Store (AVS) menjalankan serangkaian toko video yang cukup standar. Sebelum video dapat diletakkan di rak, video tersebut harus dikatalogkan dan dimasukkan ke dalam database video. Setiap pelanggan harus memiliki kartu pelanggan AVS yang valid untuk dapat menyewa video. Pelanggan menyewa video selama tiga hari sekaligus. Setiap kali pelanggan menyewa video, sistem harus memastikan bahwa dia tidak memiliki video yang terlambat. Jika demikian, video yang telah jatuh tempo harus dikembalikan dan biaya yang telah jatuh tempo harus dibayar sebelum pelanggan dapat menyewa lebih banyak video. Demikian juga jika pelanggan telah mengembalikan video yang telah lewat waktu namun belum membayar biaya keterlambatan, maka biaya tersebut harus dibayarkan sebelum video baru dapat disewa. Setiap pagi, manajer toko mencetak laporan yang mencantumkan video yang terlambat. Jika video terlambat dua hari atau lebih, manajer akan menelepon pelanggan untuk mengingatkannya agar mengembalikan video tersebut. Jika video dikembalikan dalam kondisi rusak, pengelola akan menghapusnya dari database video dan terkadang membebankan biaya kepada pelanggan.

M. Buat satu set deskripsi Use-Case terperinci untuk sistem video dalam latihan L.

N. Lakukan penelusuran verifikasi dan validasi model fungsional sistem penyimpanan video yang dijelaskan dalam latihan L dan M.

O. Gambarlah diagram Use-Case dan serangkaian diagram aktivitas untuk sistem keanggotaan gym. Ketika anggota bergabung dengan gym, mereka membayar biaya untuk jangka waktu tertentu. Sebagian besar keanggotaan adalah untuk satu tahun, tetapi keanggotaan sesingkat dua bulan tersedia. Sepanjang tahun, gym menawarkan berbagai diskon untuk harga keanggotaan reguler mereka (misalnya, dua keanggotaan dengan harga satu untuk hari Valentine). Adalah umum bagi anggota untuk membayar jumlah yang berbeda untuk masa keanggotaan yang sama. Gym ingin mengirimkan surat pengingat kepada anggota yang meminta mereka untuk memperbarui keanggotaan mereka satu bulan sebelum keanggotaan mereka berakhir. Beberapa anggota menjadi marah ketika diminta untuk memperbarui dengan tarif yang jauh lebih tinggi daripada kontrak keanggotaan asli mereka, sehingga klub ingin melacak harga yang dibayarkan sehingga manajer dapat mengganti harga reguler dengan harga khusus ketika anggota diminta untuk memperbarui. Sistem harus melacak harga baru ini sehingga pembaruan dapat diproses secara akurat. Salah satu permasalahan dalam industri ini adalah tingginya tingkat turnover anggota. Meskipun beberapa anggota tetap aktif selama bertahun-tahun, sekitar setengah dari anggota tidak memperbarui keanggotaan mereka. Ini adalah masalah besar, karena gym menghabiskan banyak iklan untuk menarik setiap anggota baru. Manajer ingin sistem untuk melacak setiap kali anggota datang ke gym. Sistem kemudian akan mengidentifikasi pengguna berat

dan membuat laporan sehingga manajer dapat meminta mereka untuk memperbarui keanggotaan mereka lebih awal, mungkin menawarkan tarif yang lebih rendah untuk perpanjangan awal. Demikian juga, sistem harus mengidentifikasi anggota yang belum mengunjungi gym selama lebih dari sebulan, sehingga manajer dapat menghubungi mereka dan mencoba untuk menarik kembali mereka ke gym.

P. Buat satu set deskripsi Use-Case terperinci untuk sistem dalam latihan O.

Q. Lakukan penelusuran verifikasi dan validasi model fungsional sistem keanggotaan gym yang dijelaskan dalam latihan O dan P.

R. Gambarlah diagram Use-Case dan serangkaian diagram aktivitas untuk sistem berikut. Picnics R Us (PRU) adalah sebuah perusahaan katering kecil dengan lima karyawan. Selama akhir pekan musim panas yang khas, PRU melayani lima belas piknik dengan masing-masing dua puluh hingga lima puluh orang. Bisnis telah berkembang pesat selama setahun terakhir, dan pemiliknya ingin memasang sistem komputer baru untuk mengelola proses pemesanan dan pembelian. PRU memiliki sepuluh menu standar. Ketika calon pelanggan menelepon, resepsionis menjelaskan menu kepada mereka. Jika pelanggan memutuskan untuk memesan piknik, resepsionis mencatat informasi pelanggan (misalnya, nama, alamat, nomor telepon) dan informasi tentang piknik (misalnya, tempat, tanggal, waktu, yang mana salah satu menu standar, total harga) pada sebuah kontrak. Pelanggan kemudian mengirim salinan kontrak melalui faks dan harus menandatangani dan mengembalikannya bersama dengan deposit (seringkali dengan kartu kredit atau kartu debit) sebelum piknik dipesan secara resmi. Sisa uang dikumpulkan saat piknik diantar. Terkadang, pelanggan menginginkan sesuatu yang istimewa (mis., Kue ulang tahun). Dalam hal ini, resepsionis mengambil informasi dan memberikannya kepada pemilik, yang menentukan biayanya; resepsionis kemudian menelepon pelanggan kembali dengan informasi harga. Terkadang pelanggan menerima harganya; di lain waktu, pelanggan meminta beberapa perubahan yang harus dikembalikan ke pemilik untuk perkiraan biaya baru. Setiap minggu, pemilik melihat-lihat piknik yang dijadwalkan untuk akhir pekan itu dan memesan persediaan (mis., Piring) dan makanan (mis., Roti, ayam) yang dibutuhkan untuk membuatnya. Pemilik ingin menggunakan sistem untuk pemasaran juga. Itu harus dapat melacak bagaimana pelanggan belajar tentang PRU dan mengidentifikasi pelanggan tetap, sehingga PRU dapat mengirimkan penawaran khusus kepada mereka. Pemiliknya juga ingin melacak piknik di mana PRU mengirim kontrak, tetapi pelanggan tidak pernah menandatangani kontrak dan benar-benar memesan piknik.

S. Buat satu set deskripsi Use-Case terperinci untuk sistem dalam latihan R.

T. Lakukan penelusuran verifikasi dan validasi model fungsional sistem katering yang dijelaskan dalam latihan R dan S.

U. Gambarlah diagram Use-Case dan serangkaian diagram aktivitas untuk sistem berikut. Of-the-Month Club (OTMC) adalah perusahaan muda inovatif yang menjual keanggotaan kepada orang-orang yang memiliki minat pada produk tertentu. Orang-orang membayar biaya keanggotaan selama satu tahun dan setiap bulan menerima produk melalui pos. Misalnya, OTMC memiliki klub c off ee-of-the-month yang mengirim anggota satu pon kopi khusus setiap bulan. OTMC saat ini memiliki enam keanggotaan (kopi, anggur, bir, cerutu, bunga, dan permainan komputer), yang masing-masing harganya berbeda. Pelanggan biasanya milik hanya satu, tetapi beberapa milik dua atau lebih. Ketika orang bergabung dengan OTMC, operator telepon mencatat nama, alamat surat, nomor telepon, alamat email, informasi kartu kredit, tanggal mulai, dan layanan keanggotaan (misalnya, kopi). Beberapa pelanggan meminta keanggotaan ganda atau tiga kali lipat (misalnya, dua pon kopi, tiga kotak bir). Keanggotaan permainan komputer beroperasi sedikit berbeda dari yang lain. Dalam hal ini, anggota juga harus memilih jenis permainan (aksi, arcade, fantasi/fiksi ilmiah, pendidikan, dll.) dan tingkat usia. OTMC berencana untuk memperluas jumlah

keanggotaan yang ditawarkan (misalnya, video game, film, mainan, keju, buah, dan sayuran), sehingga sistem perlu mengakomodasi ekspansi di masa mendatang ini. OTMC juga berencana menawarkan keanggotaan tiga bulan dan enam bulan.

V. Buat satu set deskripsi Use-Case terperinci untuk sistem dalam latihan U.
W. Lakukan penelusuran verifikasi dan validasi model fungsional sistem Klub Bulanan yang dijelaskan dalam latihan U dan V.

Minicase

1. ABC Printing adalah organisasi percetakan dan pengukiran kecil. Ketika Aurel, pemiliknya, membawa komputer ke kantor bisnis lima tahun lalu, bisnisnya sangat kecil dan sangat sederhana. Aurel dapat menggunakan sistem akuntansi berbasis PC yang murah untuk menangani kebutuhan pemrosesan informasi dasar perusahaan. Namun, seiring berjalannya waktu, bisnis telah berkembang dan pekerjaan yang dilakukan menjadi jauh lebih kompleks. Perangkat lunak akuntansi sederhana yang masih digunakan tidak lagi memadai untuk melacak banyak kesepakatan dan pengaturan canggih perusahaan dengan pelanggannya.

   Aurel memiliki empat orang staf di kantor bisnis yang akrab dengan seluk-beluk persyaratan pencatatan perusahaan. Aurel baru-baru ini bertemu dengan stafnya untuk membahas rencananya untuk menyewa sebuah perusahaan konsultan SI untuk mengevaluasi kebutuhan sistem informasi organisasi dan merekomendasikan strategi untuk meningkatkan sistem komputernya. Staf sangat antusias dengan prospek sistem baru, karena sistem saat ini menyebabkan banyak gangguan bagi mereka. Namun, tidak ada staf yang pernah melakukan hal seperti ini sebelumnya, dan mereka sedikit waspada terhadap konsultan yang akan melakukan proyek tersebut.

   Asumsikan bahwa Anda adalah seorang analis sistem di tim konsultan yang ditugaskan untuk keterlibatan ABC Printing. Pada pertemuan pertama Anda dengan staf Aurel, Anda ingin memastikan bahwa mereka memahami pekerjaan yang akan dilakukan tim Anda dan bagaimana mereka akan berpartisipasi dalam pekerjaan itu.

   a. Jelaskan, dalam istilah nonteknis yang jelas, tujuan analisis proyek.
   b. Jelaskan, dalam istilah nonteknis yang jelas, bagaimana model fungsional akan digunakan oleh tim proyek untuk memodelkan proses bisnis yang diidentifikasi. Jelaskan apa model-model ini, apa yang mereka wakili dalam sistem, dan bagaimana mereka akan digunakan oleh tim.
2. Manajemen Staf Profesional dan Ilmiah (PSSM) adalah jenis unik dari agen kepegawaian sementara. Banyak organisasi saat ini mempekerjakan karyawan teknis yang sangat terampil dalam jangka pendek, sementara untuk membantu proyek-proyek khusus atau untuk memberikan keterampilan teknis yang dibutuhkan. PSSM menegosiasikan kontrak dengan perusahaan kliennya di mana PSSM setuju untuk menyediakan staf sementara dalam kategori pekerjaan tertentu dengan biaya tertentu. Misalnya, PSSM memiliki kontrak dengan perusahaan eksplorasi minyak dan gas di mana ia setuju untuk memasok ahli geologi dengan setidaknya gelar master sebesar Rp. 9.000.000 per minggu. PSSM memiliki kontrak dengan berbagai perusahaan dan dapat menempatkan hampir semua jenis anggota staf profesional atau ilmiah, dari pemrogram komputer hingga ahli geologi hingga ahli astrofisika.

   Ketika perusahaan klien PSSM menentukan bahwa mereka akan membutuhkan karyawan profesional atau ilmiah sementara, itu mengeluarkan permintaan staf terhadap kontrak yang sebelumnya telah dinegosiasikan dengan PSSM. Ketika manajer kontrak PSSM menerima permintaan staf, nomor kontrak yang dirujuk pada permintaan staf dimasukkan ke dalam database kontrak. Dengan menggunakan

informasi dari database, manajer kontrak meninjau syarat dan ketentuan kontrak dan menentukan apakah permintaan kepegawaian itu valid. Permintaan kepegawaian berlaku jika kontrak belum berakhir, jenis karyawan profesional atau ilmiah yang diminta tercantum pada kontrak asli, dan biaya yang diminta termasuk dalam kisaran biaya yang dinegosiasikan. Jika permintaan staf tidak valid, manajer kontrak mengirimkan kembali permintaan staf kepada klien dengan surat yang menyatakan mengapa permintaan staf tidak dapat dipenuhi, dan salinan surat tersebut diajukan. Jika permintaan kepegawaian valid, manajer kontrak memasukkan permintaan kepegawaian ke dalam database permintaan kepegawaian sebagai permintaan kepegawaian yang belum diselesaikan. Permintaan kepegawaian kemudian dikirim ke departemen penempatan PSSM.

Di departemen penempatan, jenis anggota staf, pengalaman, dan kualifikasi yang diminta pada permintaan penempatan diperiksa terhadap database staf profesional dan ilmiah yang tersedia. Jika individu yang memenuhi syarat ditemukan, dia ditandai "dipesan" di database staf. Jika individu yang memenuhi syarat tidak dapat ditemukan dalam database atau tidak segera tersedia, departemen penempatan membuat memo yang menjelaskan ketidakmampuan untuk memenuhi permintaan kepegawaian dan melampirkannya ke permintaan kepegawaian. Semua permintaan staf kemudian dikirim ke departemen pengaturan.

Di bagian pengaturan, calon pegawai tidak tetap dihubungi dan diminta untuk menyetujui penempatan tersebut. Setelah rincian penempatan telah dikerjakan dan disetujui, anggota staf ditandai "ditempatkan" di database staf. Salinan permintaan staf dan tagihan untuk biaya penempatan dikirim ke klien. Terakhir, permintaan kepegawaian, memo “tidak dapat diisi” (jika ada), dan salinan tagihan biaya penempatan dikirim ke manajer kontrak. Jika permintaan kepegawaian dipenuhi, manajer kontrak menutup permintaan kepegawaian terbuka di database permintaan kepegawaian. Jika permintaan staf tidak dapat dipenuhi, klien akan diberitahu. Permintaan kepegawaian, tagihan biaya penempatan, dan memo yang tidak dapat diisi kemudian diajukan di kantor kontrak.

a. Buat diagram Use-Case untuk sistem yang dijelaskan di sini.
b. Buat satu set diagram aktivitas untuk proses bisnis yang dijelaskan di sini.
c. Untuk setiap Use-Case utama yang diidentifikasi dalam diagram Use-Case, kembangkan baik gambaran umum maupun deskripsi Use-Case yang detail.
d. Verifikasi dan validasi model fungsional.

BAB 5

PERMODELAN STRUKTURAL

Model struktural, atau konseptual, menggambarkan struktur objek yang mendukung proses bisnis dalam suatu organisasi. Selama analisis, model struktural menyajikan organisasi logis dari objek tanpa menunjukkan bagaimana mereka disimpan, dibuat, atau dimanipulasi sehingga analis dapat fokus pada bisnis, tanpa terganggu oleh detail teknis. Kemudian selama desain, model struktural diperbarui untuk mencerminkan dengan tepat bagaimana objek akan disimpan dalam database dan file. Bab ini menjelaskan kartu class-responsibility-collaboration (CRC), diagram kelas, dan diagram objek, yang digunakan untuk membuat model struktural.

5.1 Tujuan

- Memahami aturan dan pedoman gaya untuk membuat kartu CRC, diagram kelas, dan diagram objek.
- Memahami proses yang digunakan untuk membuat kartu CRC, diagram kelas, dan diagram objek.
- Mampu membuat kartu CRC, diagram kelas, dan diagram objek.
- Memahami hubungan antara model struktural.
- Memahami hubungan antara model struktural dan fungsional.

5.2 Pendahuluan

Selama analisis, analis membuat proses bisnis dan model fungsional untuk mewakili bagaimana sistem bisnis akan berperilaku. Pada saat yang sama, analis perlu memahami informasi yang digunakan dan dibuat oleh sistem bisnis (misalnya, informasi pelanggan, informasi pesanan). Dalam bab ini, kita membahas bagaimana objek yang mendasari perilaku yang dimodelkan dalam proses bisnis dan model fungsional diatur dan disajikan.

Seperti yang ditunjukkan dalam Bab 1, semua pendekatan pengembangan sistem berorientasi objek didorong oleh Use-Case, arsitektur-sentris, dan iteratif dan inkremental. Use case, yang dijelaskan dalam Bab 4, membentuk fondasi di mana sistem informasi bisnis dibuat. Dari perspektif arsitektur-sentris, pemodelan struktural mendukung penciptaan pandangan struktural atau statis internal dari sistem informasi bisnis yang menunjukkan bagaimana sistem terstruktur untuk mendukung proses bisnis yang mendasarinya. Akhirnya, seperti halnya proses bisnis dan pemodelan fungsional, Anda akan menemukan bahwa Anda tidak hanya perlu melakukan iterasi di seluruh model struktural (dijelaskan dalam bab ini), tetapi Anda juga harus melakukan iterasi di ketiga tampilan arsitektur (fungsional, struktural, dan perilaku) untuk sepenuhnya menangkap dan mewakili persyaratan untuk sistem informasi bisnis.

Model struktural adalah cara formal untuk mewakili objek yang digunakan dan dibuat oleh sistem bisnis. Ini menggambarkan orang, tempat, atau hal-hal tentang informasi mana yang ditangkap dan bagaimana mereka terkait satu sama lain. Model struktural digambar menggunakan proses berulang di mana model menjadi lebih rinci dan kurang konseptual dari waktu ke waktu. Dalam analisis, analis menggambar model konseptual, yang menunjukkan organisasi logis dari objek tanpa menunjukkan bagaimana objek disimpan, dibuat, atau dimanipulasi. Karena model ini bebas dari implementasi atau detail teknis apa pun, analis dapat lebih mudah fokus pada pencocokan model dengan kebutuhan bisnis sistem yang sebenarnya.

Dalam desain, analis mengembangkan model struktural konseptual menjadi model desain yang mencerminkan bagaimana objek akan diatur dalam database dan perangkat lunak. Pada titik ini, model diperiksa untuk redundansi, dan analis menyelidiki cara untuk membuat objek mudah diambil. Spesifik model desain dibahas secara rinci dalam bab desain.

5.3 MODEL STRUKTUR

Setiap kali seorang analis sistem menghadapi masalah baru untuk dipecahkan, analis harus mempelajari domain masalah yang mendasarinya. Tujuan analis adalah untuk menemukan objek utama yang terkandung dalam domain masalah dan untuk membangun model struktural. Pemodelan berorientasi objek memungkinkan analis untuk mengurangi kesenjangan semantik antara domain masalah yang mendasari dan model struktural berkembang. Namun, dunia nyata dan dunia perangkat lunak sangat berbeda. Dunia nyata cenderung berantakan, sedangkan dunia perangkat lunak harus rapi dan logis. Dengan demikian, pemetaan yang tepat antara model struktural dan domain masalah mungkin tidak mungkin. Bahkan, itu mungkin tidak diinginkan.

Salah satu tujuan utama dari model struktural adalah untuk menciptakan kosakata yang dapat digunakan oleh analis dan pengguna. Model struktural mewakili hal-hal, ide, atau konsep yang terkandung dalam domain masalah. Mereka juga memungkinkan representasi hubungan antara hal-hal, ide, atau konsep. Dengan membuat model struktural dari domain masalah, analis menciptakan kosakata yang diperlukan bagi analis dan pengguna untuk berkomunikasi secara efektif.

Penting untuk diingat bahwa pada tahap pengembangan ini, model struktural tidak mewakili komponen perangkat lunak atau kelas dalam bahasa pemrograman berorientasi objek, meskipun model struktural memang berisi kelas analisis, atribut, operasi, dan hubungan antara kelas analisis. Penyempurnaan kelas-kelas awal ini menjadi objek tingkat pemrograman muncul kemudian. Meskipun demikian, model struktural pada titik ini harus mewakili tanggung jawab masing-masing kelas dan kolaborasi antar kelas. Biasanya, model struktural digambarkan menggunakan kartu CRC, diagram kelas, dan, dalam beberapa kasus, diagram objek. Namun, sebelum menjelaskan kartu CRC, diagram kelas, dan diagram objek, kami menjelaskan elemen dasar model struktural: kelas, atribut, operasi, dan hubungan.

Kelas, Atribut, dan Operasi

Kelas adalah template umum yang kita gunakan untuk membuat instance, atau objek tertentu, dalam domain masalah. Semua objek dari kelas tertentu identik dalam struktur dan perilaku tetapi mengandung data yang berbeda dalam atributnya. Ada dua jenis umum kelas minat selama analisis: konkret dan abstrak. Biasanya, ketika seorang analis menjelaskan kelas domain aplikasi, dia mengacu pada kelas konkret; yaitu, kelas konkret digunakan untuk membuat objek. Kelas abstrak sebenarnya tidak ada di dunia nyata; mereka hanyalah abstraksi yang berguna. Misalnya, dari kelas karyawan dan kelas pelanggan, kami dapat mengidentifikasi generalisasi dari dua kelas dan menamai orang kelas abstrak. Kami mungkin tidak benar-benar membuat instance kelas orang dalam sistem itu sendiri, alih-alih membuat dan hanya menggunakan karyawan dan pelanggan.

Klasifikasi kelas kedua adalah jenis benda dunia nyata yang diwakili oleh kelas. Ada kelas domain, kelas antarmuka pengguna, kelas struktur data, kelas struktur file, kelas lingkungan operasi, kelas dokumen, dan berbagai jenis kelas multimedia. Pada titik ini dalam pengembangan sistem kami yang berkembang, kami hanya tertarik pada kelas domain. Kemudian dalam desain dan implementasi, jenis kelas lain menjadi lebih relevan.

Atribut kelas analisis mewakili sepotong informasi yang relevan dengan deskripsi kelas dalam domain aplikasi dari masalah yang sedang diselidiki. Atribut berisi informasi yang menurut analis atau pengguna harus dilacak oleh sistem. Misalnya, atribut yang mungkin relevan dari kelas karyawan adalah nama karyawan, sedangkan atribut yang mungkin tidak relevan adalah warna rambut. Keduanya menggambarkan sesuatu tentang seorang karyawan, tetapi warna rambut mungkin tidak terlalu berguna untuk sebagian besar aplikasi bisnis. Hanya atribut yang penting untuk tugas yang harus disertakan dalam kelas. Terakhir, hanya atribut yang merupakan tipe primitif atau atomik (yaitu, bilangan bulat, string, ganda, tanggal, waktu, Boolean, dll.) yang harus ditambahkan. Atribut yang paling kompleks atau majemuk benar-benar pengganti untuk hubungan antar kelas. Oleh karena itu, mereka harus dimodelkan sebagai hubungan, bukan sebagai atribut (lihat bagian selanjutnya).

Perilaku kelas analisis didefinisikan dalam operasi atau layanan. Pada fase selanjutnya, operasi diubah menjadi metode. Namun, karena metode lebih terkait dengan implementasi, pada titik ini dalam pengembangan kami menggunakan istilah operasi untuk menggambarkan tindakan yang dapat ditanggapi oleh instance kelas. Seperti atribut, hanya operasi spesifik domain masalah yang relevan dengan masalah yang diselidiki yang harus dipertimbangkan. Misalnya, biasanya diperlukan bahwa kelas menyediakan sarana untuk membuat instance, menghapus instance, mengakses nilai atribut individual, menetapkan nilai atribut individual, mengakses nilai hubungan individual, dan menghapus nilai hubungan individual. Namun, pada titik ini dalam pengembangan sistem yang berkembang, analis harus menghindari mengacaukan definisi kelas dengan jenis operasi dasar ini dan hanya fokus pada operasi spesifik domain masalah yang relevan.

Hubungan

Ada banyak jenis hubungan yang dapat didefinisikan, tetapi semuanya dapat diklasifikasikan ke dalam tiga kategori dasar mekanisme abstraksi data: hubungan generalisasi, hubungan agregasi, dan hubungan asosiasi. Mekanisme abstraksi data ini memungkinkan analis untuk fokus pada dimensi penting sementara mengabaikan dimensi yang tidak penting. Seperti halnya atribut, analis harus berhati-hati untuk memasukkan hanya hubungan yang relevan.

Hubungan Generalisasi. Abstraksi generalisasi memungkinkan analis untuk membuat kelas yang mewarisi atribut dan operasi dari kelas lain. Analis membuat superclass yang berisi atribut dasar dan operasi yang akan digunakan di beberapa subclass. Subclass mewarisi atribut dan operasi superclass mereka dan juga dapat berisi atribut dan operasi yang unik hanya untuk mereka. Misalnya, kelas pelanggan dan kelas karyawan dapat digeneralisasi menjadi kelas orang dengan mengekstraksi atribut dan operasi yang sama-sama dimiliki dan menempatkannya ke dalam superkelas baru, orang. Dengan cara ini, analis dapat mengurangi redundansi dalam definisi kelas sehingga elemen umum didefinisikan sekali dan kemudian digunakan kembali di subkelas. Generalisasi direpresentasikan dengan hubungan sejenis, sehingga kita mengatakan bahwa seorang karyawan adalah orang yang sejenis.

Analis juga dapat menggunakan kebalikan dari generalisasi. Spesialisasi mengungkap kelas tambahan dengan mengizinkan subkelas baru dibuat dari kelas yang ada. Misalnya, kelas karyawan dapat dispesialisasikan menjadi kelas sekretaris dan kelas insinyur.

Selanjutnya, hubungan generalisasi antar kelas dapat digabungkan untuk membentuk hierarki generalisasi. Berdasarkan contoh sebelumnya, kelas sekretaris dan kelas insinyur dapat menjadi subkelas dari kelas karyawan, yang pada gilirannya dapat menjadi subkelas dari kelas orang. Ini akan dibaca sebagai sekretaris dan seorang insinyur adalah karyawan yang baik hati dan pelanggan dan seorang karyawan adalah orang yang baik hati.

Abstraksi data generalisasi adalah mekanisme yang sangat kuat yang mendorong analis untuk fokus pada properti yang membuat setiap kelas unik dengan memungkinkan kesamaan untuk difaktorkan ke dalam superclass. Namun, untuk memastikan bahwa semantik subclass dipertahankan, analis harus menerapkan prinsip substitusi. Maksud kami bahwa subkelas harus mampu menggantikan superkelas di mana saja yang menggunakan superkelas (misalnya, di mana pun kami menggunakan superkelas karyawan, kami juga dapat secara logis menggunakan subkelas sekretarisnya). Dengan berfokus pada interpretasi sejenis dari hubungan generalisasi, prinsip substitusi diterapkan.

Hubungan Agregasi. Secara umum, semua hubungan agregasi menghubungkan bagian dengan keseluruhan atau rakitan. Untuk tujuan kami, kami menggunakan hubungan semantik a-part-of atau has-parts untuk mewakili abstraksi agregasi. Misalnya, pintu adalah bagian dari mobil, karyawan adalah bagian dari departemen, atau departemen adalah bagian dari organisasi. Seperti hubungan generalisasi, hubungan agregasi dapat digabungkan menjadi hierarki agregasi. Misalnya, piston adalah bagian dari mesin, dan mesin adalah bagian dari mobil.

Hubungan agregasi bersifat dua arah. Sisi lain dari agregasi adalah dekomposisi. Analis dapat menggunakan dekomposisi untuk mengungkap bagian dari kelas yang harus dimodelkan secara terpisah. Misalnya, jika pintu dan mesin adalah bagian dari mobil, maka mobil memiliki bagian pintu dan mesin. Analis dapat melompat-lompat di antara berbagai bagian untuk mengungkap bagian-bagian baru. Misalnya, analis dapat bertanya, Apa bagian lain dari mobil? atau Untuk rakitan lain mana sebuah pintu dapat dimiliki?

Hubungan Asosiasi. Ada jenis hubungan lain yang tidak cocok dengan kerangka generalisasi (a-kind-of) atau agregasi (a-part-of). Secara teknis, hubungan ini biasanya merupakan bentuk yang lebih lemah dari hubungan agregasi. Misalnya, pasien menjadwalkan janji temu. Dapat dikatakan bahwa pasien adalah bagian dari janji. Namun, ada perbedaan semantik yang jelas antara jenis hubungan ini dan hubungan yang memodelkan hubungan antara pintu dan mobil atau bahkan pekerja dan serikat pekerja. Dengan demikian, mereka hanya dianggap sebagai asosiasi antara instance kelas.

5.4 IDENTIFIKASI OBYEK

Pendekatan yang berbeda telah disarankan untuk membantu analis dalam mengidentifikasi satu set objek kandidat untuk model struktural. Empat pendekatan yang paling umum adalah analisis tekstual, brainstorming, daftar objek umum, dan pola. Sebagian besar analis menggunakan kombinasi teknik ini untuk memastikan bahwa tidak ada objek penting dan atribut objek, operasi, dan hubungan yang diabaikan.

Analisis Tekstual

Analis melakukan analisis tekstual dengan meninjau diagram use-case dan memeriksa teks dalam deskripsi use-case untuk mengidentifikasi objek potensial, atribut, operasi, dan hubungan. Kata benda dalam use case menyarankan kemungkinan kelas, dan kata kerja menyarankan kemungkinan operasi. Gambar 5-1 menyajikan ringkasan panduan yang berguna. Analisis tekstual deskripsi use-case telah dikritik karena terlalu sederhana, tetapi karena tujuan utamanya adalah untuk membuat model struktural awal yang kasar, kesederhanaannya merupakan keuntungan utama. Sebagai contoh, jika kita menerapkan aturan ini pada use case Make Old Patient Appt yang dijelaskan pada Bab 4 dan direplikasi pada Gambar 5-2, kita dapat dengan mudah mengidentifikasi objek potensial untuk pasien lama, dokter, janji temu, pasien, kantor, resepsionis, nama , alamat, informasi pasien, pembayaran, tanggal, dan waktu. Kami juga dapat dengan mudah mengidentifikasi operasi potensial yang dapat dikaitkan dengan objek yang diidentifikasi. Misalnya, pasien

menghubungi kantor, membuat janji baru, membatalkan janji yang sudah ada, mengubah janji yang sudah ada, mencocokkan waktu dan tanggal janji yang diminta dengan waktu dan tanggal yang diminta, dan menemukan janji saat ini.

- Kata benda umum atau tidak tepat menyiratkan kelas objek.
- Sebuah kata benda yang tepat atau referensi langsung menyiratkan sebuah instance dari kelas.
- Kata benda kolektif menyiratkan kelas objek yang terdiri dari kelompok-kelompok instance dari kelas lain.
- Kata sifat menyiratkan atribut suatu objek.
- Kata kerja melakukan menyiratkan operasi.
- Sebuah kata kerja menjadi menyiratkan hubungan klasifikasi antara objek dan kelasnya.
- Kata kerja yang memiliki menyiratkan hubungan agregasi atau asosiasi.
- Kata kerja transitif menyiratkan operasi.
- Kata kerja intransitif menyiratkan pengecualian.
- Sebuah predikat atau frase verba deskriptif menyiratkan suatu operasi.
- Kata keterangan menyiratkan atribut dari suatu hubungan atau operasi.

Diadaptasi dari Russell J. Abbott, "Program Design by Informal English Descriptions," Communications of the ACM 26, no. 11 (1983): 882–894; Peter P-S Chen, "Struktur Kalimat Bahasa Inggris dan Diagram Hubungan Entitas," Ilmu Informasi: Jurnal Internasional 29, no. 2-3 (1983): 127–149; Ian Graham, Teknologi Migrasi ke Objek (Reading, MA: Addison Wesley Longman, 1995).

Gambar 5-1 Pedoman Analisis Tekstual

Brainstorming

Brainstorming adalah teknik penemuan yang telah berhasil digunakan dalam mengidentifikasi kelas kandidat. Pada dasarnya, dalam konteks ini, brainstorming adalah proses di mana sekelompok individu yang duduk di sekitar meja menyarankan kelas potensial yang dapat berguna untuk masalah yang sedang dipertimbangkan. Biasanya, sesi curah pendapat dimulai oleh seorang fasilitator yang meminta sekelompok individu untuk menjawab pertanyaan atau pernyataan spesifik yang membingkai sesi tersebut. Misalnya, dengan menggunakan masalah janji temu yang dijelaskan sebelumnya, fasilitator dapat meminta tim pengembangan dan pengguna untuk memikirkan pengalaman mereka membuat janji dan mengidentifikasi kelas kandidat berdasarkan pengalaman masa lalu mereka. Perhatikan bahwa pendekatan ini tidak menggunakan model fungsional yang dikembangkan sebelumnya. Ini hanya meminta peserta untuk mengidentifikasi objek yang mereka telah berinteraksi. Misalnya, sekumpulan objek potensial yang muncul dalam pikiran adalah dokter, perawat, resepsionis, janji temu, penyakit, pengobatan, resep, kartu asuransi, dan catatan medis. Setelah jumlah kandidat objek yang cukup telah diidentifikasi, para peserta harus mendiskusikan dan memilih objek kandidat mana yang harus dipertimbangkan lebih lanjut. Setelah ini telah diidentifikasi, brainstorming lebih lanjut dapat dilakukan untuk mengidentifikasi atribut potensial, operasi, dan hubungan untuk masing-masing objek yang diidentifikasi.

Pertama, semua saran harus ditanggapi dengan serius. Pada titik ini dalam pengembangan sistem, jauh lebih baik untuk menghapus sesuatu lebih lambat daripada secara tidak sengaja meninggalkan sesuatu yang penting. Kedua, semua peserta harus mulai berpikir cepat dan marah. Setelah semua ide keluar di atas meja pepatah, maka para peserta dapat didorong untuk merenungkan kelas kandidat yang telah mereka identifikasi. Ketiga, fasilitator

harus mengatur proses berpikir cepat dan marah. Jika tidak, prosesnya akan kacau. Selanjutnya, fasilitator harus memastikan bahwa semua peserta terlibat dan bahwa beberapa peserta tidak mendominasi proses. Untuk mendapatkan gambaran masalah yang paling lengkap, kami menyarankan menggunakan pendekatan round-robin di mana peserta bergiliran menyarankan kelas kandidat. Pendekatan lain adalah dengan menggunakan alat brainstorming elektronik yang mendukung anonimitas. Keempat, fasilitator dapat menggunakan humor untuk mencairkan suasana sehingga semua peserta merasa nyaman dalam memberikan saran.

Nama Use Case : Buat Aplikasi Pasien Lama	ID : 2	Level Kesulitan : High
Primary Actor: Pasien Lama	Use-Case Type: Detail, Essential	
Stakeholders and Interests: Pasien Lama – ingin membuat, mengubah, atau membatalkan janji Dokter – ingin memastikan kebutuhan pasien terpenuhi tepat waktu		
Deskripsi Singkat: Kasus penggunaan ini menjelaskan bagaimana kami membuat janji temu serta mengubah atau membatalkan janji temu untuk pasien yang sebelumnya terlihat.		
Pemicu: Pasien menelepon dan meminta janji baru atau meminta untuk membatalkan atau mengubah janji yang sudah ada. Type : External		
Relationships: Association: Sales Rep Include: Extend: Generalization:		
Alur Peristiwa Normal: 1. Pasien menghubungi kantor mengenai janji temu. 2. Pasien memberikan Resepsionis dengan nama dan alamatnya. 3. Jika informasi Pasien telah berubah Jalankan kasus penggunaan Perbarui Informasi Pasien. 4. Jika pengaturan pembayaran Pasien telah berubah, Jalankan use case Lakukan Pengaturan Pembayaran. 5. Resepsionis bertanya kepada Pasien apakah dia ingin membuat janji baru, membatalkan janji yang sudah ada, atau mengubah janji yang sudah ada. Jika pasien ingin membuat janji baru, subflow S-1: janji baru dilakukan. Jika pasien ingin membatalkan janji temu yang ada, subflow S-2: batalkan janji temu dilakukan. Jika pasien ingin mengubah janji temu yang sudah ada, subflow S-3: ubah janji temu dilakukan.		

6. Resepsionis memberikan hasil transaksi kepada Pasien.
Subflow: S-1: Janji Baru 1. Resepsionis menanyakan kemungkinan waktu janji temu kepada Pasien. 2. Resepsionis mencocokkan waktu janji temu yang diinginkan Pasien dengan tanggal dan waktu yang tersedia serta menjadwalkan janji temu baru. S-2: Batalkan Janji Temu 1. Resepsionis menanyakan waktu janji lama kepada Pasien. 2. Resepsionis menemukan janji temu saat ini dalam file janji temu dan membatalkannya. S-3: Ubah Janji Temu 1. Resepsionis melakukan subflow S-2: batalkan janji temu. 2. Resepsionis melakukan S-1: subflow janji baru
Aliran Alternatif/Luar Biasa: S-1, 2a1: Resepsionis mengusulkan beberapa alternatif waktu janji temu berdasarkan yang tersedia di jadwal janji temu. S-1, 2a2: Pasien memilih salah satu waktu yang diusulkan atau memutuskan untuk tidak membuat janji.

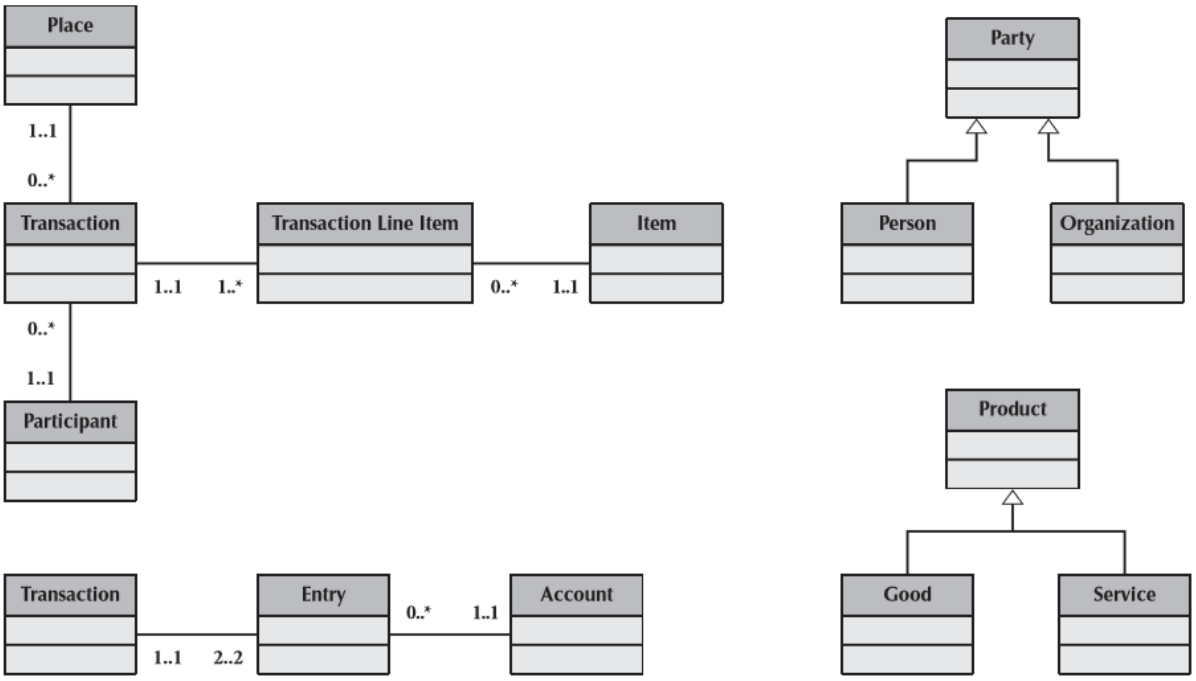
Gambar 5-2 Deskripsi Use-Case (Gambar 4-13)

Daftar Objek Umum

Seperti namanya, daftar objek umum hanyalah daftar objek yang umum untuk domain bisnis sistem. Beberapa kategori objek telah ditemukan untuk membantu analis dalam membuat daftar, seperti hal-hal fisik atau nyata, insiden, peran, dan interaksi. Analis pertama-tama harus mencari hal-hal fisik, atau nyata, dalam domain bisnis. Ini dapat mencakup buku, meja, kursi, dan peralatan kantor. Biasanya, jenis objek ini adalah yang paling mudah untuk diidentifikasi. Insiden adalah peristiwa yang terjadi dalam domain bisnis, seperti rapat, penerbangan, pertunjukan, atau kecelakaan. Meninjau Use-Case dapat dengan mudah mengidentifikasi peran yang dimainkan orang dalam masalah, seperti dokter, perawat, pasien, atau resepsionis. Biasanya, interaksi adalah transaksi yang terjadi di domain bisnis, seperti transaksi penjualan. Jenis objek lain yang dapat diidentifikasi termasuk tempat, wadah, organisasi, catatan bisnis, katalog, dan kebijakan. Dalam kasus yang jarang terjadi, proses itu sendiri mungkin memerlukan informasi yang disimpan tentang mereka. Dalam kasus ini, proses mungkin memerlukan objek, selain use case, untuk mewakilinya. Terakhir, ada perpustakaan objek yang dapat digunakan kembali yang telah dibuat untuk domain bisnis yang berbeda. Misalnya, sehubungan dengan masalah janji temu, Objek Medis Sumber Terbuka Umum5 dapat berguna untuk menyelidiki objek potensial yang harus disertakan.

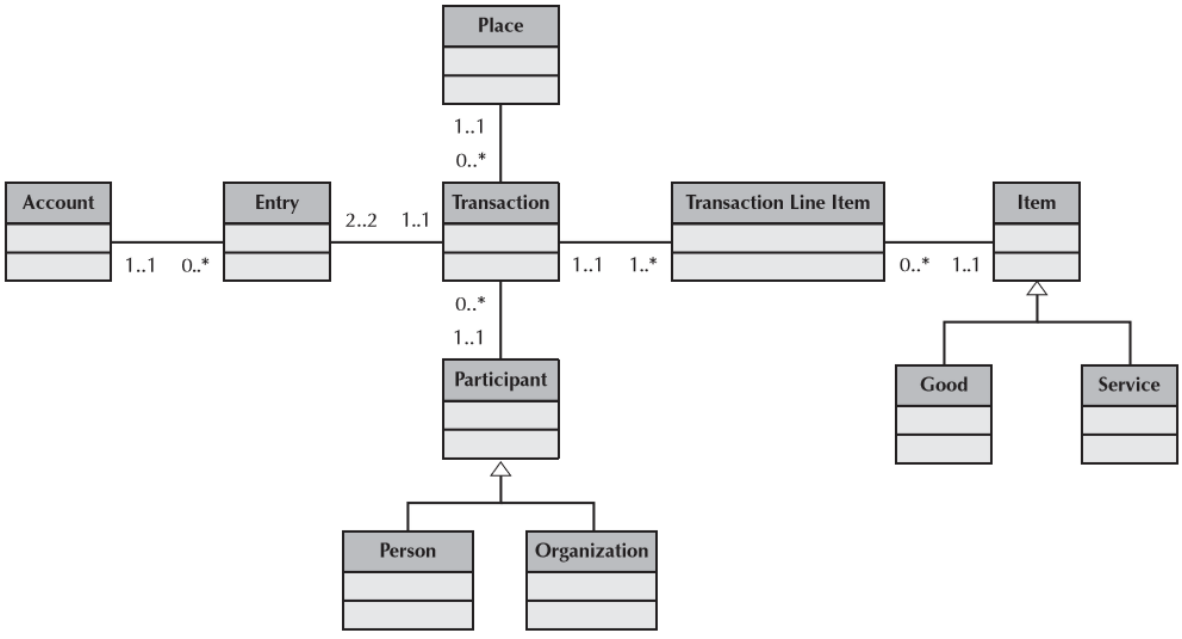
Pola

Ide menggunakan pola adalah area yang relatif baru dalam pengembangan sistem berorientasi objek.6 Ada banyak definisi tentang apa itu pola. Dari sudut pandang kami, sebuah pola hanyalah sekelompok kelas yang berkolaborasi yang memberikan solusi untuk masalah yang umum terjadi. Karena pola memberikan solusi untuk masalah yang sering terjadi, pola dapat digunakan kembali.



Gambar 5-3 Contoh Pola

Seorang arsitek, sebut saja Alexander, telah mengilhami banyak pekerjaan yang terkait dengan penggunaan pola dalam pengembangan sistem berorientasi objek. Menurut Alexander dan rekan-rekannya, adalah mungkin untuk membuat bangunan yang sangat canggih dengan merangkai pola yang umum ditemukan, daripada menciptakan konsep dan desain yang sama sekali baru. Dengan cara yang sama, dimungkinkan untuk mengumpulkan pola berorientasi objek yang umum ditemukan untuk membentuk sistem informasi berorientasi objek yang elegan. Misalnya, banyak transaksi bisnis melibatkan jenis objek dan interaksi yang sama. Hampir semua transaksi akan memerlukan kelas transaksi, kelas item baris transaksi, kelas item, kelas lokasi, dan kelas peserta. Dengan menggunakan kembali pola kelas yang ada, kita dapat lebih cepat dan lebih lengkap mendefinisikan sistem daripada jika kita mulai dengan selembar kertas kosong.



Gambar 5-4 Contoh Integrasi Pola Sampel

Banyak jenis pola telah diusulkan, mulai dari pola berorientasi bisnis tingkat tinggi hingga pola desain tingkat rendah. Misalnya, Gambar 5-3 menggambarkan satu set pola analisis yang berguna. Gambar 5-4 menggambarkan diagram kelas yang kita buat dengan menggabungkan pola yang terdapat pada Gambar 5-3 menjadi satu pola yang dapat digunakan kembali. Dalam hal ini, kami menggabungkan pola Transaksi– Masuk–Akun (terletak di kiri bawah Gambar 5-3) dengan pola Tempat–Transaksi–Peserta–Transaksi Item Baris–Item (terletak di kiri atas Gambar 5-3) pada kelas Transaksi umum. Selanjutnya, kami menggabungkan Party–Person–Organization (terletak di kanan atas Gambar 5-3) dengan menggabungkan kelas Participant dan Party. Terakhir, kami memperluas kelas Item dengan menggabungkan kelas Item dengan kelas Produk dari pola Product–Good–Service (terletak di kanan bawah Gambar 5-3).

Dengan cara ini, menggunakan pola dari sumber yang berbeda dalam memungkinkan tim pengembangan untuk meningkatkan pengetahuan di luar anggota tim langsung dan memungkinkan tim untuk mengembangkan model domain masalah yang lebih lengkap dan kuat. Misalnya, dalam kasus masalah janji temu, kita dapat melihat objek yang sebelumnya diidentifikasi melalui analisis tekstual, curah pendapat, dan/atau daftar objek umum dan melihat apakah masuk akal untuk memetakan salah satu dari mereka ke dalam pola yang dapat digunakan kembali yang telah ditentukan sebelumnya. Dalam kasus khusus ini, kita dapat melihat janji temu sebagai jenis transaksi di mana kantor dokter berpartisipasi. Dengan melihat janji temu sebagai jenis transaksi, kita dapat menerapkan pola yang kita buat pada Gambar 5-4 dan menemukan sekumpulan objek yang sebelumnya tidak teridentifikasi, seperti Tempat, Pasien sebagai jenis Peserta, dan Item Baris Transaksi yang terkait dengan berbagai jenis Barang (Barang dan/atau Jasa). Menemukan objek tambahan khusus ini dapat berguna dalam mengembangkan sisi penagihan dari sistem janji temu. Meskipun objek tambahan ini dapat diterapkan, mereka tidak ditemukan menggunakan teknik lain.

Business Domains	Sources of Patterns
Accounting	3, 4
Actor-Role	2
Assembly-Part	1
Container-Content	1
Contract	2, 4
Document	2, 4
Employment	2, 4
Financial Derivative Contracts	3
Geographic Location	2, 4
Group-Member	1
Interaction	1
Material Requirements Planning	4
Organization and Party	2, 3
Plan	1, 3
Process Manufacturing	4
Trading	3
Transactions	1, 4

Gambar 5-5 Pola Berguna

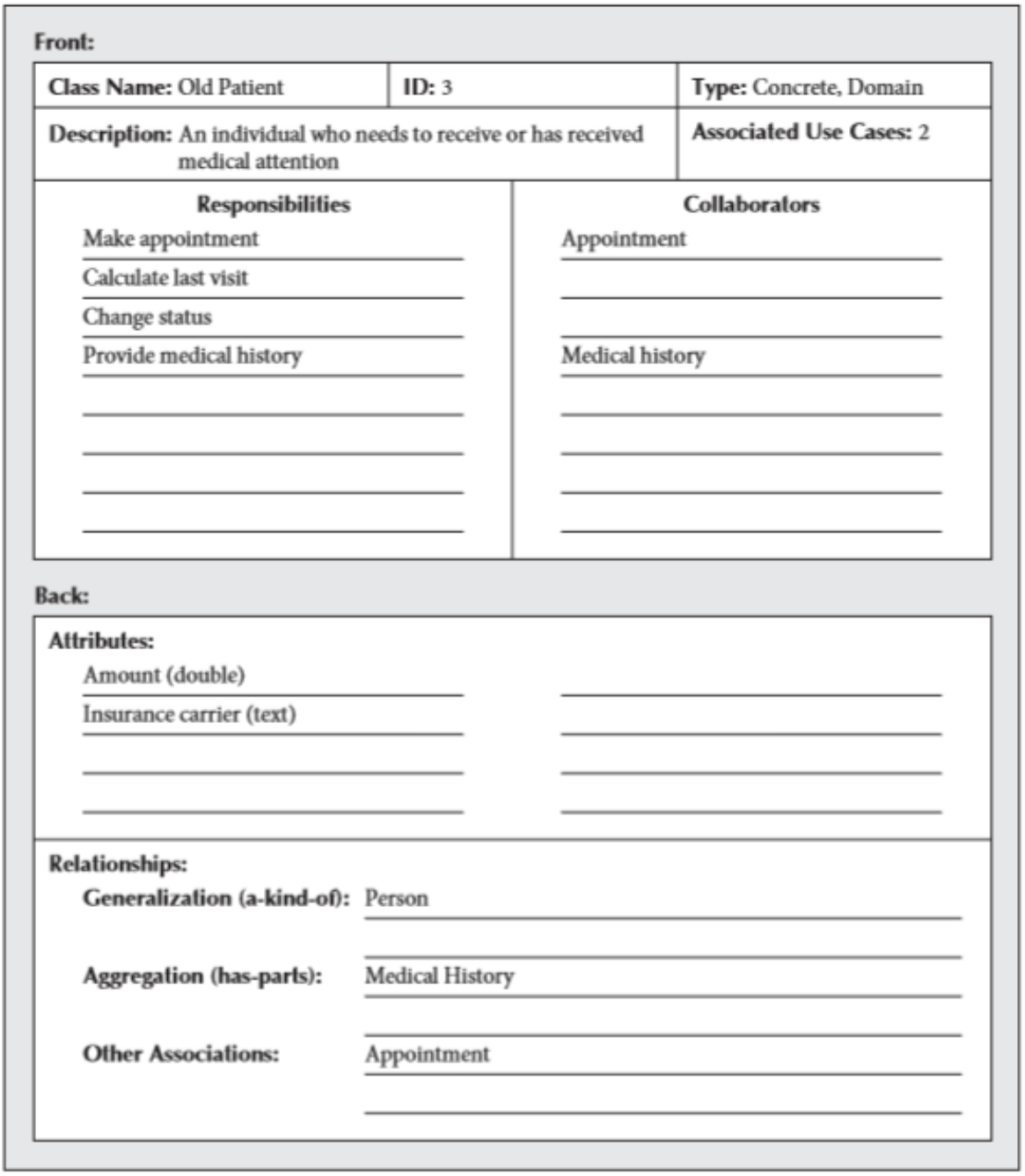
Berdasarkan contoh sederhana ini, jelas bahwa menggunakan pola untuk mengembangkan model struktural dapat menguntungkan. Gambar 5-5 mencantumkan beberapa domain bisnis umum yang polanya telah dikembangkan dan sumbernya. Jika kita mengembangkan sistem informasi bisnis di salah satu domain bisnis ini, maka pola yang dikembangkan untuk domain tersebut mungkin menjadi titik awal yang sangat berguna dalam mengidentifikasi kelas yang dibutuhkan dan atribut, operasi, dan hubungannya.

5.5 KARTU CRC

Kartu CRC (Class–Responsibility–Collaboration) digunakan untuk mendokumentasikan tanggung jawab dan kolaborasi kelas. Dalam beberapa metodologi pengembangan sistem berorientasi objek, kartu CRC dipandang sebagai pesaing alternatif untuk penggunaan Use-Case dan diagram kelas Proses Terpadu. Namun, kami melihatnya sebagai pendekatan berteknologi rendah yang berguna yang dapat melengkapi pendekatan Proses Terpadu berteknologi tinggi yang menggunakan CASE tool. Kami menggunakan bentuk kartu CRC yang diperluas untuk menangkap semua informasi relevan yang terkait dengan suatu kelas. Kami menjelaskan elemen kartu CRC kami nanti, setelah kami menjelaskan tanggung jawab dan kolaborasi.

Tanggung Jawab dan Kerjasama

Tanggung jawab kelas dapat dibagi menjadi dua jenis terpisah: mengetahui dan melakukan. Mengetahui tanggung jawab adalah hal-hal yang harus dapat diketahui oleh instance kelas. Sebuah instance dari sebuah kelas biasanya mengetahui nilai dari atributnya dan hubungannya. Melakukan tanggung jawab adalah hal-hal yang harus mampu dilakukan oleh instance kelas. Dalam hal ini, sebuah instance dari suatu kelas dapat menjalankan operasinya atau dapat meminta instance kedua, yang diketahuinya, untuk mengeksekusi salah satu operasinya atas nama instance pertama.



Gambar 5-6 Contoh Kartu CRC

Model struktural menggambarkan objek yang diperlukan untuk mendukung proses bisnis yang dimodelkan oleh Use-Case. Sebagian besar Use-Case melibatkan sekumpulan beberapa kelas, bukan hanya satu kelas.

Kelas-kelas ini membentuk kolaborasi. Kolaborasi memungkinkan analis untuk berpikir dalam hal klien, server, dan kontrak. Objek klien adalah turunan dari kelas yang mengirimkan permintaan ke turunan kelas lain untuk menjalankan operasi. Objek server adalah instance yang menerima permintaan dari objek klien. Kontrak memformalkan interaksi antara objek klien dan server. Bab 8 memberikan penjelasan yang lebih rinci tentang kontrak dan contoh penggunaannya.

Seorang analis dapat menggunakan gagasan tanggung jawab kelas dan kolaborasi kontrak klien-server untuk membantu mengidentifikasi kelas, bersama dengan atribut, operasi, dan hubungan, yang terlibat dengan Use-Case. Salah satu cara termudah untuk menggunakan kartu CRC dalam mengembangkan model struktural adalah melalui antropomorfisme — berpura-pura bahwa kelas memiliki karakteristik manusia. Anggota tim pengembangan dapat mengajukan pertanyaan kepada diri mereka sendiri atau ditanyai oleh anggota tim lainnya. Biasanya pertanyaan yang diajukan berupa:

Siapa atau apa kamu?

Apa yang Anda tahu?

Apa yang bisa kau lakukan?

Jawaban atas pertanyaan tersebut kemudian digunakan untuk menambahkan detail pada kartu CRC yang berkembang. Misalnya, dalam masalah janji temu, seorang anggota tim dapat berpura-pura bahwa dia adalah seorang janji temu. Dalam hal ini, janji akan menjawab bahwa dia tahu tentang dokter dan pasien yang berpartisipasi dalam janji dan mereka akan tahu tanggal dan waktu janji. Lebih jauh lagi, janji temu harus mengetahui cara membuat dirinya sendiri, menghapus dirinya sendiri, dan kemungkinan mengubah aspek yang berbeda dari dirinya sendiri. Dalam beberapa kasus, pendekatan ini akan mengungkap objek tambahan yang harus ditambahkan ke model struktural yang berkembang.

Elemen Kartu CRC

Himpunan kartu CRC berisi semua informasi yang diperlukan untuk membangun model struktural logis dari masalah yang sedang diselidiki. Gambar 5-6 menunjukkan contoh kartu CRC. Setiap kartu CRC menangkap dan menjelaskan elemen penting dari sebuah kelas. Bagian depan kartu berisi nama kelas, ID, jenis, deskripsi, Use-Case terkait, tanggung jawab, dan kolaborator. Nama kelas harus berupa kata benda (tetapi bukan kata benda yang tepat, seperti nama orang atau benda tertentu). Sama seperti Use-Case, pada tahap pengembangan selanjutnya, penting untuk dapat melacak kembali keputusan desain ke persyaratan tertentu. Dalam hubungannya dengan daftar Use-Case terkait, nomor ID untuk setiap kelas dapat digunakan untuk mencapai hal ini. Deskripsi hanyalah pernyataan singkat yang dapat digunakan sebagai definisi tekstual untuk kelas. Tanggung jawab kelas cenderung menjadi operasi yang harus dikandung kelas (yaitu, tanggung jawab melakukan).

Bagian belakang kartu CRC berisi atribut dan hubungan kelas. Atribut kelas mewakili tanggung jawab yang harus dipenuhi oleh setiap instance kelas. Biasanya, tipe data setiap atribut terdaftar dengan nama atribut (misalnya, atribut jumlah ganda dan pembawa asuransi adalah teks). Tiga jenis hubungan biasanya ditangkap pada titik ini: generalisasi, agregasi, dan asosiasi lainnya. Pada Gambar 5-6, kita melihat bahwa Pasien adalah orang yang sejenis dan Pasien dikaitkan dengan Janji Temu.

Kartu CRC digunakan untuk mendokumentasikan properti penting dari suatu kelas. Namun, setelah kartu diisi, analis dapat menggunakan kartu dan antropomorfisme dalam permainan peran (dijelaskan di bagian berikutnya) untuk mengungkap properti yang hilang dengan menjalankan skenario berbeda yang terkait dengan Use-Case (lihat Bab 4). Role-playing juga dapat digunakan sebagai dasar untuk menguji kejelasan dan kelengkapan representasi sistem yang berkembang.

Kartu CRC Bermain Peran dengan Use Cases

Selain pendekatan identifikasi objek yang dijelaskan sebelumnya (analisis tekstual, brainstorming, daftar objek umum, dan pola), kartu CRC dapat digunakan dalam latihan

permainan peran yang telah terbukti berguna dalam menemukan objek tambahan, atribut, hubungan, dan operasi. Lebih lanjut, selain penelusuran, yang dijelaskan kemudian dalam bab ini, permainan peran sangat berguna dalam menguji kesetiaan model struktural yang berkembang. Secara umum, anggota tim melakukan peran yang terkait dengan aktor dan objek yang sebelumnya diidentifikasi dengan Use-Case yang berbeda. Secara teknis, anggota tim melakukan langkah-langkah berbeda yang terkait dengan skenario spesifik dari use case. Ingat, skenario adalah jalur eksekusi tunggal yang unik melalui use case. Tempat yang berguna untuk mencari skenario yang berbeda dari use case adalah diagram aktivitas (misalnya, lihat Gambar 4-8, 4-9, 4-10, dan 4-12). Skenario yang berbeda ada untuk setiap kali simpul keputusan menyebabkan perpecahan di jalur eksekusi Use-Case. Selain itu, skenario dapat diidentifikasi dari aliran alternatif/luar biasa dalam deskripsi Use-Case. Mempertimbangkan sifat inkremental dan iteratif dan bahwa diagram aktivitas dan deskripsi Use-Case harus berisi informasi yang sama, meninjau kedua representasi akan memastikan bahwa skenario yang relevan tidak terlewatkan.

Langkah pertama adalah meninjau deskripsi use-case (lihat Gambar 5-2). Ini memungkinkan tim untuk memilih Use-Case tertentu untuk bermain peran. Meskipun tergoda untuk mencoba menyelesaikan Use-Case sebanyak mungkin dalam waktu singkat, tim sebaiknya tidak memilih Use-Case yang paling mudah terlebih dahulu. Sebaliknya, pada titik ini dalam pengembangan sistem, tim harus memilih use case yang paling penting, paling kompleks, atau paling tidak dipahami.

Langkah kedua adalah mengidentifikasi peran relevan yang akan dimainkan. Setiap peran dikaitkan dengan aktor atau objek. Untuk memilih objek yang relevan, tim meninjau setiap kartu CRC dan memilih yang terkait dengan Use-Case yang dipilih. Sebagai contoh, pada Gambar 5-6, kita melihat bahwa kartu CRC yang mewakili kelas Old Patient dikaitkan dengan Use Case nomor 2. Jadi jika kita akan memainkan role-play use case Make Old Patient Appt (lihat Gambar 5- 2), kita perlu menyertakan kartu CRC Pasien Lama. Dengan meninjau deskripsi Use-Case, kita dapat dengan mudah mengidentifikasi aktor Pasien dan Dokter Lama (lihat bagian Aktor Utama dan Pemegang kepentingan dari deskripsi Use-Case pada Gambar 5-2). Dengan membaca bagian acara dari deskripsi Use-Case, kami mengidentifikasi peran aktor internal Resepsionis. Setelah mengidentifikasi semua peran yang relevan, kami menetapkan masing-masing peran tersebut kepada anggota tim yang berbeda.

Langkah ketiga adalah memainkan skenario Use-Case dengan meminta anggota tim melakukan masing-masing skenario. Untuk melakukan ini, setiap anggota tim harus berpura-pura bahwa dia adalah contoh dari peran yang diberikan kepadanya. Misalnya, jika seorang anggota tim diberi peran sebagai Resepsionis, maka dia harus dapat melakukan langkah-langkah berbeda dalam skenario yang terkait dengan Resepsionis. Dalam hal skenario perubahan penunjukan, ini akan mencakup langkah 2, 5, 6, S-3, S-1, dan S-2. Namun, ketika skenario ini dilakukan (role-played), akan ditemukan bahwa langkah 1, 3, dan 4 tidak lengkap. Misalnya, pada Langkah 1, apa yang sebenarnya terjadi? Apakah Pasien melakukan panggilan telepon? Jika demikian, siapa yang menjawab telepon? Dengan kata lain, banyak informasi yang terkandung dalam deskripsi use-case hanya diidentifikasi secara implisit, bukan eksplisit. Ketika informasi tidak diidentifikasi secara eksplisit, ada banyak ruang untuk interpretasi, yang mengharuskan anggota tim untuk membuat asumsi. Jauh lebih baik untuk menghilangkan kebutuhan untuk membuat asumsi dengan membuat setiap langkah eksplisit. Dalam hal ini, Langkah 1 dari Alur Peristiwa Normal harus dimodifikasi. Setelah langkah diperbaiki, skenario dicoba lagi. Proses ini dilakukan berulang-ulang sampai skenario dapat dieksekusi sampai pada kesimpulan yang berhasil. Setelah skenario berhasil diselesaikan, skenario berikutnya dilakukan. Ini diulang sampai semua skenario use case dapat dilakukan dengan sukses.

Langkah keempat adalah dengan mengulang langkah 1 sampai 3 untuk Use-Case yang tersisa.

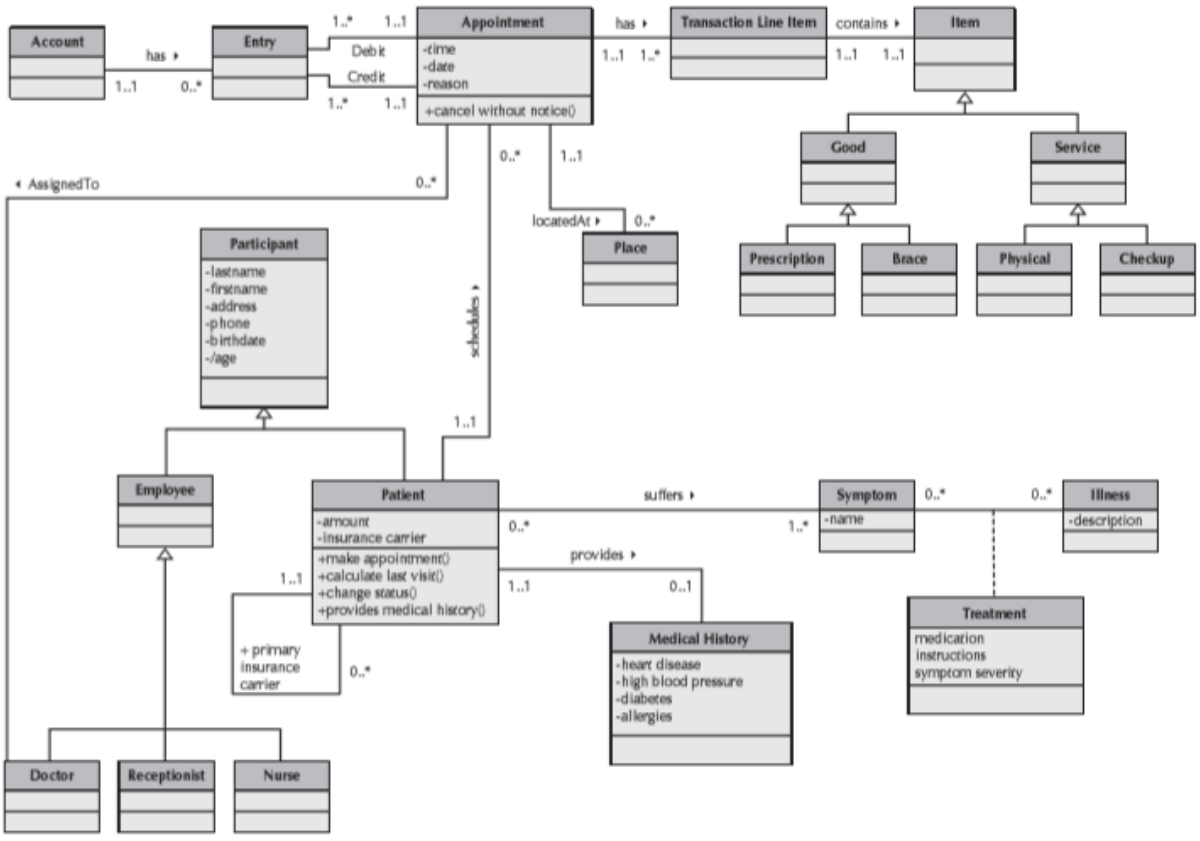
5.6 DIAGRAM KELAS

Diagram kelas adalah model statis yang menunjukkan kelas dan hubungan antar kelas yang tetap konstan dalam sistem dari waktu ke waktu. Diagram kelas menggambarkan kelas, yang mencakup perilaku dan status, dengan hubungan antar kelas. Bagian berikut menyajikan elemen diagram kelas, pendekatan berbeda yang dapat digunakan untuk menyederhanakan diagram kelas, dan diagram struktur alternatif: diagram objek.

Elemen Diagram Kelas

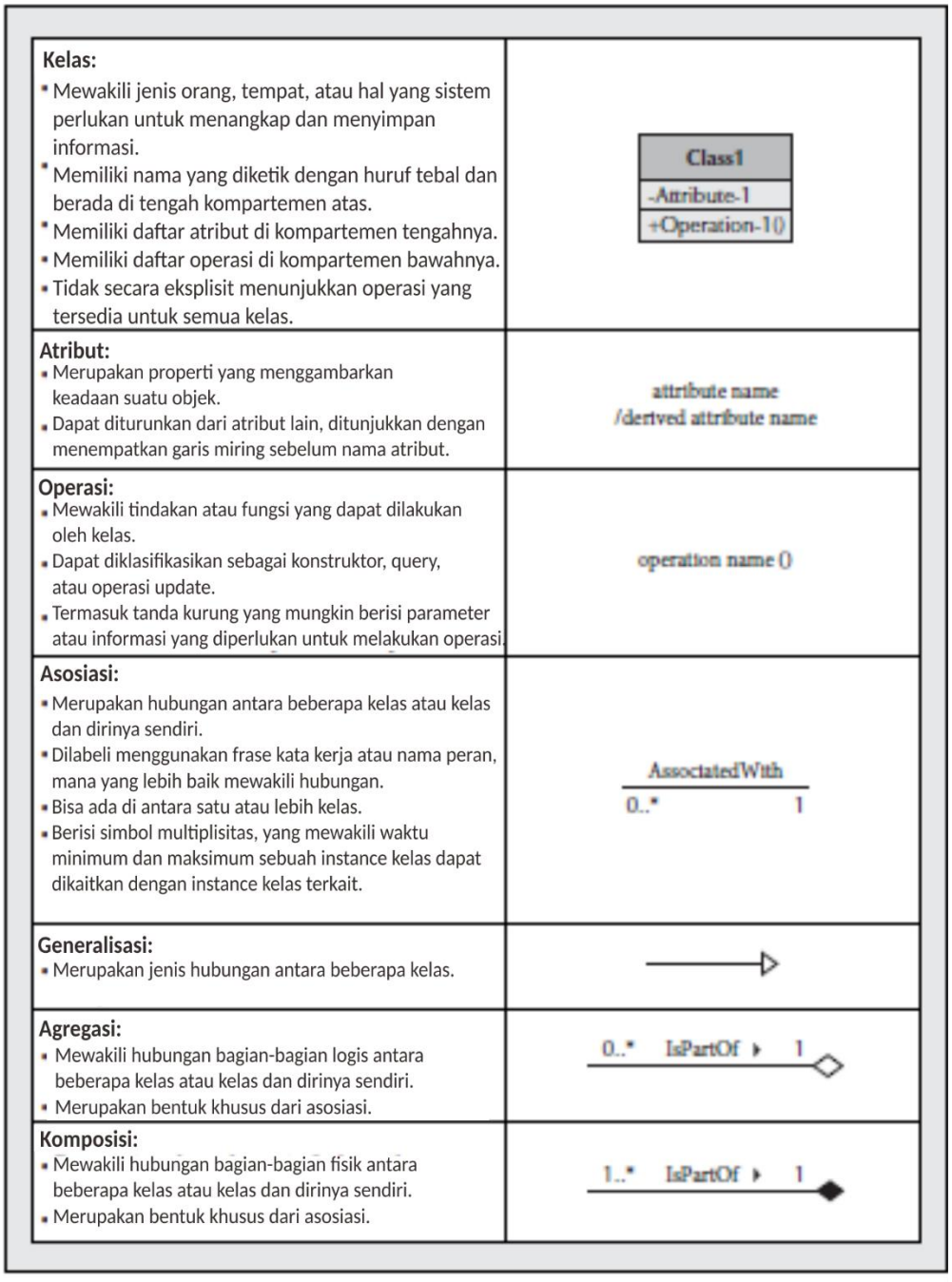
Gambar 5-7 menunjukkan diagram kelas yang dibuat untuk mencerminkan kelas dan hubungan yang terkait dengan sistem penunjukan. Diagram ini didasarkan pada kelas-kelas yang ditemukan melalui teknik identifikasi objek dan permainan peran kartu CRC yang dijelaskan sebelumnya.

Kelas. Blok bangunan utama dari diagram kelas adalah kelas, yang menyimpan dan mengelola informasi dalam sistem (lihat Gambar 5-8). Selama analisis, kelas mengacu pada orang, tempat, dan hal-hal tentang sistem yang akan menangkap informasi. Kemudian, selama desain dan implementasi, kelas dapat merujuk ke artefak khusus implementasi seperti jendela, formulir, dan objek lain yang digunakan untuk membangun sistem. Setiap kelas digambar menggunakan persegi panjang tiga bagian, dengan nama kelas di atas, atribut di tengah, dan operasi di bawah. Kita dapat melihat bahwa kelas-kelas yang telah diidentifikasi sebelumnya, seperti Peserta, Dokter, Pasien, Resepsionis, Riwayat Kesehatan, Pengangkatan, dan Gejala, tercakup dalam Gambar 5-7. Atribut kelas dan nilainya menentukan keadaan setiap objek yang dibuat dari kelas, dan perilaku diwakili oleh operasi.



Gambar 5-7 Contoh Diagram Kelas

Atribut adalah properti dari kelas yang ingin kita tangkap informasinya (lihat Gambar 5-8). Perhatikan bahwa kelas Participant pada Gambar 5-7 berisi atribut: nama belakang, nama depan, alamat, telepon, dan tanggal lahir. Terkadang, Anda mungkin ingin menyimpan atribut turunan, yang merupakan atribut yang dapat dihitung atau diturunkan; atribut khusus ini dilambangkan dengan menempatkan garis miring (/) sebelum nama atribut. Perhatikan bagaimana kelas orang berisi atribut turunan yang disebut/usia, yang dapat diturunkan dengan mengurangkan tanggal lahir pasien dari tanggal saat ini. Dimungkinkan juga untuk menunjukkan visibilitas atribut pada diagram. Visibilitas berkaitan dengan tingkat penyembunyian informasi yang akan diterapkan untuk atribut tersebut. Visibilitas suatu atribut dapat berupa public (+), protected (#), atau private (−). Atribut publik adalah atribut yang tidak disembunyikan dari objek lain. Dengan demikian, objek lain dapat mengubah nilainya. Atribut yang dilindungi adalah atribut yang disembunyikan dari semua kelas lain kecuali subkelas langsungnya. Atribut privat adalah atribut yang disembunyikan dari semua kelas lainnya. Visibilitas default untuk suatu atribut biasanya bersifat pribadi.



Kelas: ▪ Mewakili jenis orang, tempat, atau hal yang sistem perlukan untuk menangkap dan menyimpan informasi. ▪ Memiliki nama yang diketik dengan huruf tebal dan berada di tengah kompartemen atas. ▪ Memiliki daftar atribut di kompartemen tengahnya. ▪ Memiliki daftar operasi di kompartemen bawahnya. ▪ Tidak secara eksplisit menunjukkan operasi yang tersedia untuk semua kelas.	Class1 -Attribute-1 +Operation-1()
Atribut: ▪ Merupakan properti yang menggambarkan keadaan suatu objek. ▪ Dapat diturunkan dari atribut lain, ditunjukkan dengan menempatkan garis miring sebelum nama atribut.	attribute name /derived attribute name
Operasi: ▪ Mewakili tindakan atau fungsi yang dapat dilakukan oleh kelas. ▪ Dapat diklasifikasikan sebagai konstruktor, query, atau operasi update. ▪ Termasuk tanda kurung yang mungkin berisi parameter atau informasi yang diperlukan untuk melakukan operasi.	operation name ()
Asosiasi: ▪ Merupakan hubungan antara beberapa kelas atau kelas dan dirinya sendiri. ▪ Dilabeli menggunakan frase kata kerja atau nama peran, mana yang lebih baik mewakili hubungan. ▪ Bisa ada di antara satu atau lebih kelas. ▪ Berisi simbol multiplisitas, yang mewakili waktu minimum dan maksimum sebuah instance kelas dapat dikaitkan dengan instance kelas terkait.	AssociatedWith 0..* 1
Generalisasi: ▪ Merupakan jenis hubungan antara beberapa kelas.	
Agregasi: ▪ Mewakili hubungan bagian-bagian logis antara beberapa kelas atau kelas dan dirinya sendiri. ▪ Merupakan bentuk khusus dari asosiasi.	0..* IsPartOf 1
Komposisi: ▪ Mewakili hubungan bagian-bagian fisik antara beberapa kelas atau kelas dan dirinya sendiri. ▪ Merupakan bentuk khusus dari asosiasi.	1..* IsPartOf 1

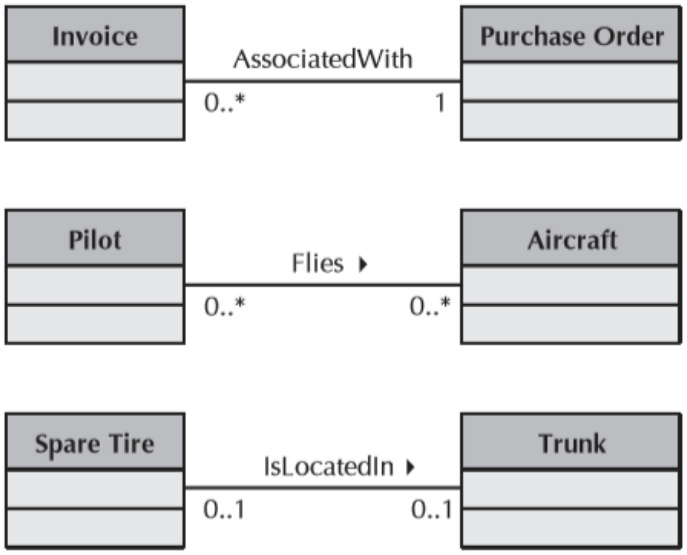
Gambar 5-8 Sintaks Diagram Kelas

Operasi adalah tindakan atau fungsi yang dapat dilakukan oleh kelas (lihat Gambar 5-8). Fungsi yang tersedia untuk semua kelas (misalnya, membuat instance baru, mengembalikan nilai untuk atribut tertentu, menetapkan nilai untuk atribut tertentu, menghapus instance) tidak secara eksplisit ditampilkan dalam persegi panjang kelas. Sebagai gantinya, hanya operasi unik untuk kelas yang disertakan, seperti operasi pembatalan tanpa pemberitahuan di kelas Appointment dan operasi hitung kunjungan terakhir di kelas Pasien pada Gambar 5-7. Perhatikan bahwa kedua operasi diikuti oleh tanda kurung, yang berisi parameter yang diperlukan oleh operasi. Jika operasi tidak memiliki parameter, tanda kurung tetap ditampilkan tetapi kosong. Seperti halnya atribut, visibilitas operasi dapat ditetapkan sebagai publik, dilindungi, atau pribadi. Visibilitas default untuk operasi biasanya bersifat publik.

Ada empat jenis operasi yang dapat berisi kelas: konstruktor, query, update, dan destruktor. Operasi konstruktor membuat instance baru dari sebuah kelas. Misalnya, kelas pasien mungkin memiliki metode yang disebut insert(), yang membuat instance pasien baru saat pasien dimasukkan ke dalam sistem. Seperti yang baru saja kita sebutkan, jika suatu operasi mengimplementasikan salah satu fungsi dasar (misalnya, membuat instance baru), biasanya tidak secara eksplisit ditampilkan pada diagram kelas, jadi biasanya kita tidak melihat metode konstruktor secara eksplisit pada diagram kelas.

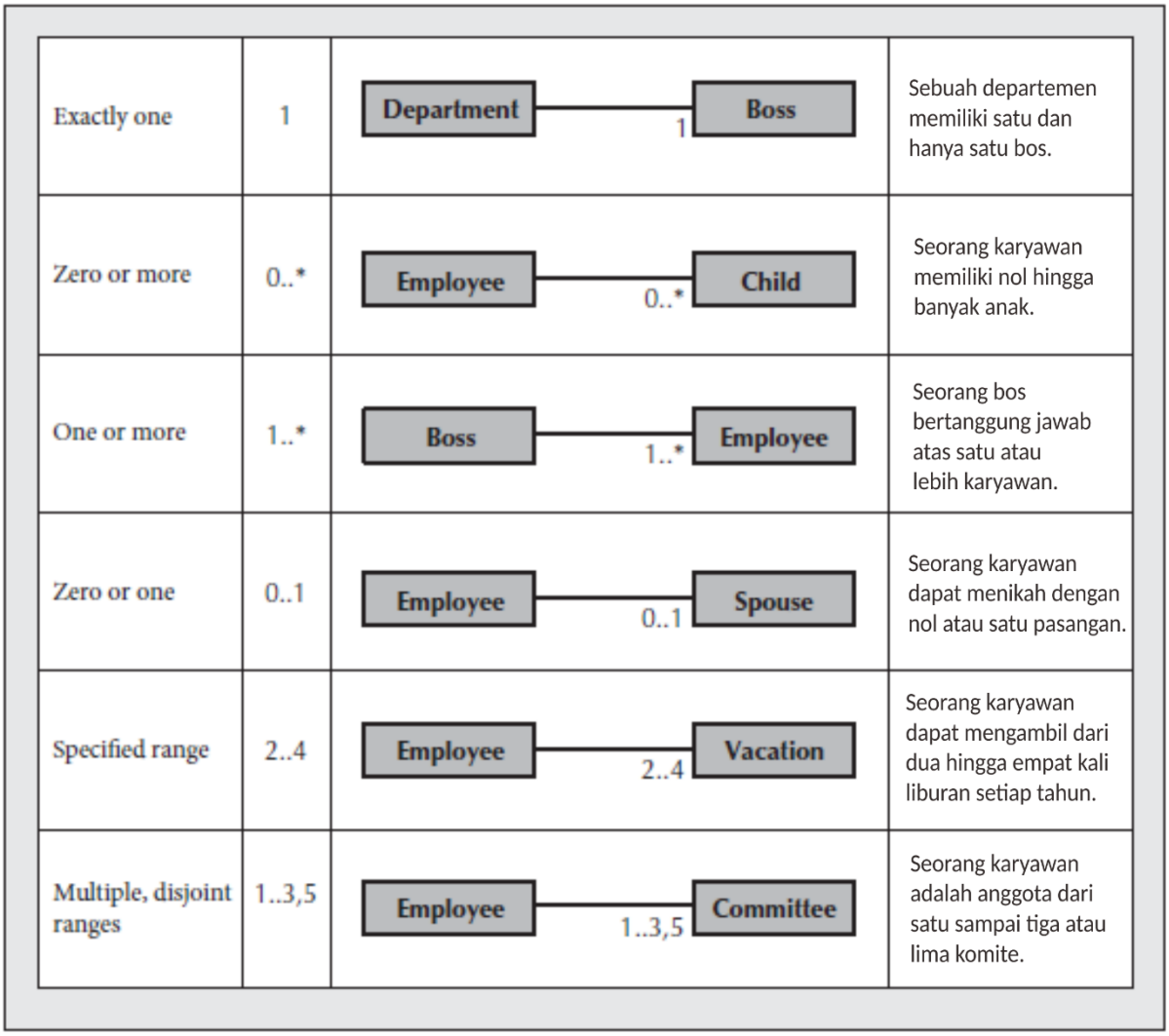
Operasi kueri membuat informasi tentang status objek tersedia untuk objek lain, tetapi tidak mengubah objek dengan cara apa pun. Misalnya, operasi hitung kunjungan terakhir () yang menentukan kapan pasien terakhir mengunjungi kantor dokter akan mengakibatkan objek diakses oleh sistem, tetapi tidak akan mengubah informasinya. Jika metode query hanya meminta informasi dari atribut di kelas (misalnya, nama pasien, alamat, telepon), maka tidak ditampilkan pada diagram karena kita menganggap bahwa semua objek memiliki operasi yang menghasilkan nilai atribut mereka.

Operasi pembaruan mengubah nilai beberapa atau semua atribut objek, yang dapat mengakibatkan perubahan status objek. Pertimbangkan untuk mengubah status pasien dari baru ke saat ini dengan metode yang disebut change status() atau mengaitkan pasien dengan janji temu tertentu dengan membuat janji temu (appointment). Jika hasil operasi dapat mengubah keadaan objek, maka operasi tersebut harus dicantumkan secara eksplisit pada diagram kelas. Di sisi lain, jika operasi pembaruan adalah operasi penugasan sederhana, itu dapat dihilangkan dari diagram.



Gambar 5-9 Contoh Asosiasi

Operasi destruktor hanya menghapus atau menghapus objek dari sistem. Misalnya, jika objek karyawan tidak lagi mewakili karyawan sebenarnya yang terkait dengan perusahaan, karyawan tersebut mungkin perlu dihapus dari database karyawan, dan operasi destruktor akan digunakan untuk mengimplementasikan perilaku ini. Namun, menghapus objek adalah salah satu fungsi dasar dan karena itu tidak akan disertakan pada diagram kelas.



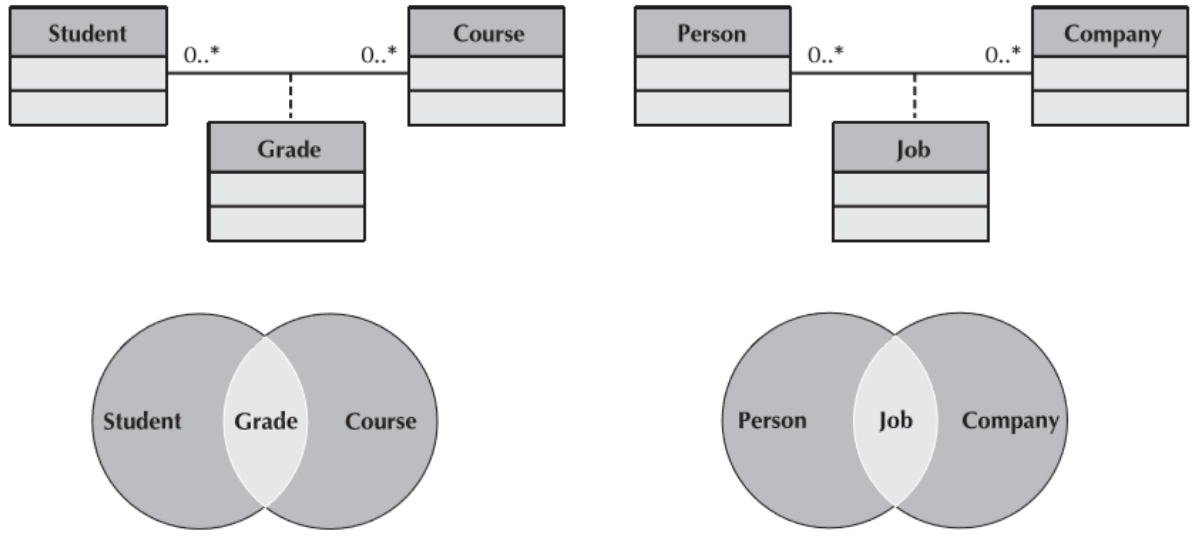
Gambar 5-10 Multiplisitas

Hubungan. Tujuan utama dari diagram kelas adalah untuk menunjukkan hubungan, atau asosiasi, yang dimiliki kelas satu sama lain. Ini digambarkan pada diagram dengan menggambar garis antar kelas (lihat Gambar 5-8). Ketika beberapa kelas berbagi hubungan (atau kelas berbagi hubungan dengan dirinya sendiri), sebuah garis digambar dan diberi label dengan nama hubungan atau peran yang dimainkan kelas dalam hubungan. Misalnya, pada Gambar 5-7 dua kelas pasien dan janji temu dikaitkan satu sama lain setiap kali pasien menjadwalkan janji temu. Dengan demikian, garis berlabel jadwal menghubungkan pasien dan janji temu, mewakili dengan tepat bagaimana kedua kelas terkait satu sama lain. Juga, perhatikan bahwa ada segitiga kecil di samping nama hubungan. Segitiga memungkinkan arah untuk dikaitkan dengan nama hubungan. Pada Gambar 5-7, hubungan jadwal mencakup segitiga, yang menunjukkan bahwa hubungan tersebut harus dibaca sebagai "janji janji temu jadwal pasien." Pencantuman segitiga hanya meningkatkan keterbacaan diagram. Pada Gambar 5-9, tiga contoh tambahan dari asosiasi digambarkan: Faktur Terkait Dengan Pesanan

Pembelian (dan sebaliknya), Pilot Menerbangkan Pesawat, dan Ban Cadangan Ditempatkan Di Bagasi.

Kadang-kadang kelas terkait dengan dirinya sendiri, seperti dalam kasus pasien menjadi pembawa asuransi utama untuk pasien lain (misalnya, pasangan, anak-anak). Pada Gambar 5-7, perhatikan bahwa garis ditarik antara kelas pasien dan dirinya sendiri dan disebut operator asuransi utama untuk menggambarkan peran yang dimainkan kelas dalam hubungan tersebut. Perhatikan bahwa tanda plus (+) ditempatkan sebelum label untuk mengkomunikasikan bahwa itu adalah peran yang bertentangan dengan nama hubungan. Saat melabeli asosiasi, kami menggunakan nama relasi atau nama peran (bukan keduanya), mana pun yang mengomunikasikan pemahaman model yang lebih menyeluruh.

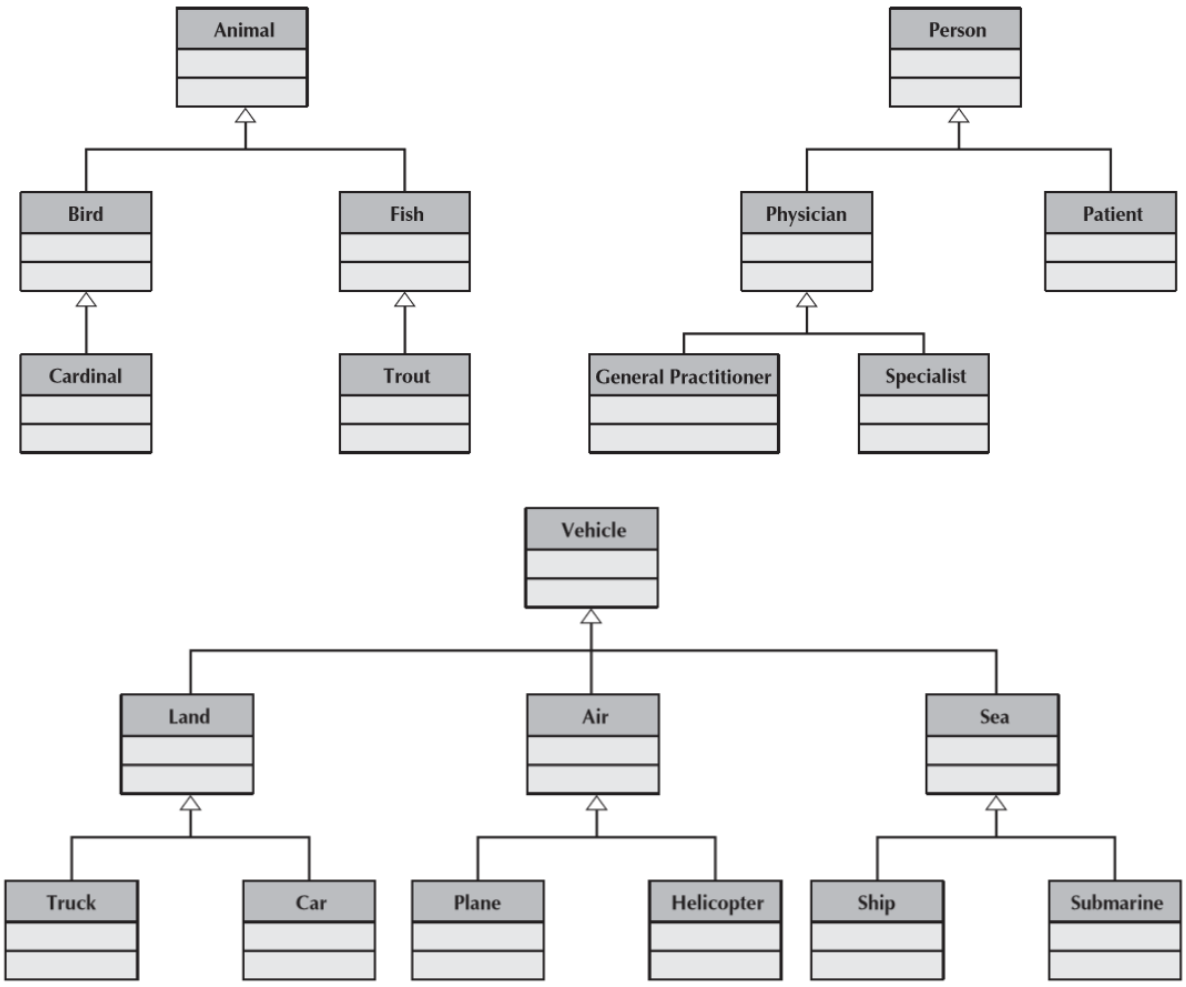
Relasi juga memiliki multiplisitas, yang mendokumentasikan bagaimana sebuah instance dari suatu objek dapat diasosiasikan dengan instance lainnya. Angka ditempatkan pada jalur asosiasi untuk menunjukkan contoh minimum dan maksimum yang dapat dihubungkan melalui asosiasi dalam format angka minimum.. angka maksimum (lihat Gambar 5-10). Angka-angka menentukan hubungan dari kelas di ujung garis hubungan sampai akhir dengan nomor. Misalnya, pada Gambar 5-7, terdapat 0..* pada akhir janji temu dari hubungan janji temu jadwal pasien. Ini berarti bahwa seorang pasien dapat dikaitkan dengan nol melalui banyak janji yang berbeda. Di akhir pasien dari hubungan yang sama ini, ada 1..1, artinya janji harus dikaitkan dengan satu dan hanya satu pasien. Pada Gambar 5-9, kita melihat bahwa instance kelas Faktur harus AssociatedWith satu instance dari kelas Purchase Order dan bahwa instance dari kelas Purchase Order mungkin AssociatedWith nol atau lebih instance dari kelas Invoice, bahwa instance dari kelas Pilot Menerbangkan nol atau lebih instance dari kelas Pesawat, dan bahwa instance kelas Pesawat dapat diterbangkan oleh nol atau lebih instance dari kelas Pilot. Akhirnya, kita melihat bahwa sebuah instance kelas Ban Cadangan IsLocatedIn nol atau satu instance dari kelas Trunk, sedangkan sebuah instance dari kelas Trunk dapat berisi nol atau satu instance dari kelas Ban Cadangan.



Gambar 5-11 Contoh Kelas Asosiasi

Ada kalanya relasi itu sendiri memiliki properti terkait, terutama saat kelasnya berbagi relasi banyak-ke-banyak. Dalam kasus ini, kelas yang disebut kelas asosiasi terbentuk, yang memiliki atribut dan operasinya sendiri. Itu ditampilkan sebagai persegi panjang yang dilampirkan oleh garis putus-putus ke jalur asosiasi, dan nama persegi panjang cocok dengan label asosiasi. Pikirkan tentang kasus menangkap informasi tentang penyakit dan gejala. Suatu penyakit (misalnya, flu) dapat dikaitkan dengan banyak gejala (misalnya, sakit tenggorokan,

demam), dan gejala (misalnya, sakit tenggorokan) dapat dikaitkan dengan banyak penyakit (misalnya, flu, radang tenggorokan, flu biasa).). Gambar 5-7 menunjukkan bagaimana kelas asosiasi dapat menangkap informasi tentang perbaikan yang berubah tergantung pada berbagai kombinasi. Misalnya, sakit tenggorokan yang disebabkan oleh radang tenggorokan memerlukan antibiotik, sedangkan pengobatan untuk sakit tenggorokan akibat flu atau pilek bisa berupa pelega tenggorokan atau teh panas. Cara lain untuk memutuskan kapan menggunakan kelas asosiasi adalah ketika atribut milik persimpangan dua kelas yang terlibat dalam asosiasi harus ditangkap. Kita dapat secara visual berpikir tentang kelas asosiasi sebagai diagram Venn. Misalnya, pada Gambar 5-11, ide Nilai benar-benar merupakan persimpangan kelas Siswa dan Kursus, karena nilai hanya ada di persimpangan dua ide ini. Contoh lain yang ditunjukkan pada Gambar 5-11 adalah bahwa pekerjaan dapat dilihat sebagai persimpangan antara Orang dan Perusahaan. Paling sering, kelas terkait melalui asosiasi normal; namun, ada dua kasus khusus dari asosiasi yang akan sering Anda lihat: generalisasi dan agregasi.



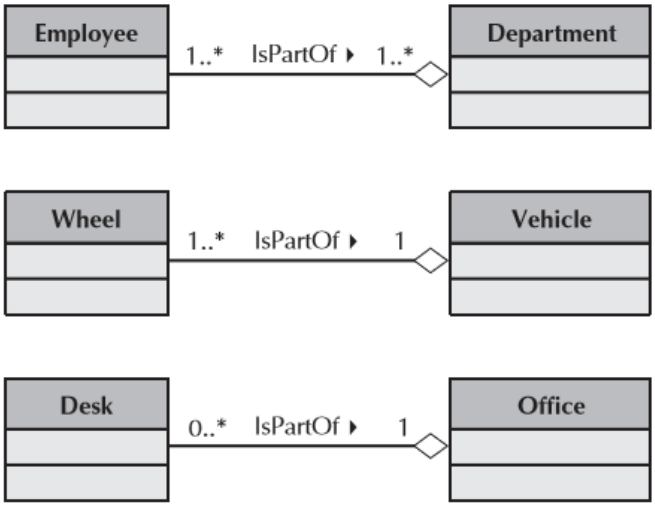
Gambar 5-12 Contoh Generalisasi

Asosiasi Generalisasi dan Agregasi. Sebuah asosiasi generalisasi menunjukkan bahwa satu kelas (subclass) mewarisi dari kelas lain (superclass), yang berarti bahwa properti dan operasi dari superclass juga berlaku untuk objek dari subclass. Jalur generalisasi ditunjukkan dengan garis padat dari subclass ke superclass dan panah berongga menunjuk ke superclass (lihat Gambar 5-8). Misalnya, Gambar 5-7 mengomunikasikan bahwa dokter, perawat, dan resepsionis adalah semua jenis karyawan dan karyawan serta pasien tersebut adalah jenis peserta. Ingatlah bahwa hubungan generalisasi terjadi ketika Anda perlu menggunakan kata-kata seperti "adalah sejenis" untuk menggambarkan hubungan tersebut.

Beberapa contoh generalisasi tambahan diberikan pada Gambar 5-12. Misalnya, Cardinal adalah sejenis Burung, yang merupakan sejenis Hewan; seorang Dokter Umum adalah Dokter yang sejenis, yang merupakan Pribadi yang sejenis; dan Truk adalah sejenis Kendaraan Darat, yaitu sejenis Kendaraan.

Asosiasi agregasi digunakan ketika kelas benar-benar terdiri dari kelas lain. Misalnya, pikirkan tentang kantor dokter yang telah memutuskan untuk membuat tim perawatan kesehatan yang mencakup dokter, perawat, dan tenaga administrasi. Saat pasien memasuki kantor, mereka ditugaskan ke tim perawatan kesehatan, yang memperhatikan kebutuhan mereka selama kunjungan mereka. Kita dapat memasukkan pengetahuan baru ini pada Gambar 5-7 dengan menambahkan dua kelas baru (Tenaga Administratif dan Tim Kesehatan) dan hubungan agregasi dari kelas Dokter, Perawat, dan Personil Administrasi baru ke kelas Tim Kesehatan yang baru. Sebuah berlian ditempatkan paling dekat dengan kelas yang mewakili agregasi (tim perawatan kesehatan), dan garis ditarik dari berlian untuk menghubungkan kelas yang berfungsi sebagai bagiannya (dokter, perawat, dan tenaga administrasi). Biasanya, Anda dapat mengidentifikasi jenis asosiasi ini ketika Anda perlu menggunakan kata-kata seperti "adalah bagian dari" atau "terdiri dari" untuk menggambarkan hubungan tersebut. Namun, dari perspektif UML, ada dua jenis asosiasi agregasi: agregasi dan komposisi (lihat Gambar 5-8).

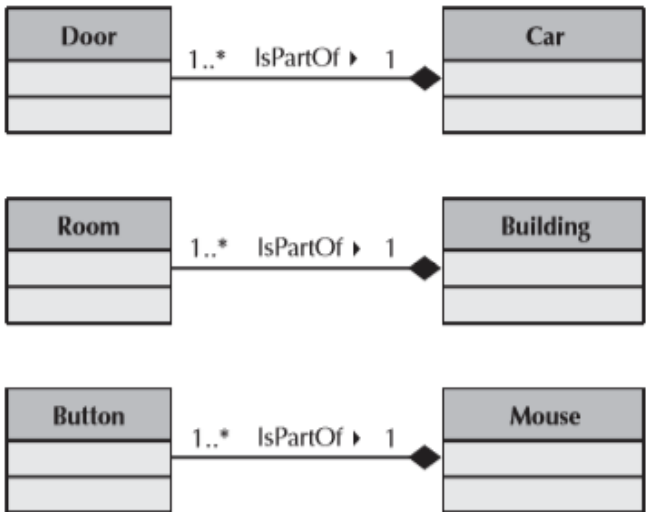
Agregasi digunakan untuk menggambarkan hubungan bagian-bagian logis dan digambarkan pada diagram kelas UML oleh berlian berongga atau putih. Misalnya pada Gambar 5-13, tiga agregasi logis ditampilkan. Logis menyiratkan bahwa adalah mungkin bagi suatu bagian untuk dikaitkan dengan banyak keutuhan atau yang relatif sederhana untuk bagian yang akan dihapus dari keseluruhan. Sebagai contoh, sebuah instance dari kelas Employee IsPartOf sebuah instance dari setidaknya satu instance dari kelas Department, sebuah instance dari kelas Wheel IsPartOf sebuah instance dari kelas Vehicle, dan sebuah instance dari kelas Desk IsPartOf sebuah instance dari kelas Office . Jelas, dalam banyak kasus seorang karyawan dapat dikaitkan dengan lebih dari satu departemen, dan relatif mudah untuk melepas roda dari kendaraan atau memindahkan meja dari kantor.



Gambar 5-13 Contoh Asosiasi Agregasi

Komposisi digunakan untuk menggambarkan bagian fisik dari hubungan dan ditunjukkan dengan berlian hitam. Fisik menyiratkan bahwa bagian dapat dikaitkan dengan hanya satu keseluruhan. Misalnya pada Gambar 5-14, tiga komposisi fisik diilustrasikan: sebuah instance pintu dapat menjadi bagian dari hanya satu instance mobil, sebuah instance dari sebuah ruangan dapat menjadi bagian dari instance hanya dari satu bangunan, dan sebuah

instance dari sebuah tombol dapat menjadi bagian dari satu mouse saja. Namun, dalam banyak kasus, perbedaan yang dapat Anda capai dengan memasukkan agregasi (berlian putih) dan komposisi (berlian hitam) dalam diagram kelas mungkin tidak sebanding dengan harga penambahan notasi grafis tambahan untuk dipelajari klien. Oleh karena itu, banyak ahli UML melihat dimasukkannya notasi agregasi dan komposisi ke diagram kelas UML hanya sebagai "gula sintaksis" dan tidak perlu karena informasi yang sama selalu dapat digambarkan hanya dengan menggunakan sintaks asosiasi.



Gambar 5-14 Contoh Komposisi Asosiasi

Menyederhanakan Diagram Kelas

Ketika diagram kelas terisi penuh dengan semua kelas dan hubungan untuk sistem dunia nyata, diagram kelas bisa menjadi sangat sulit untuk ditafsirkan (yaitu, bisa sangat kompleks). Ketika ini terjadi, terkadang diagram perlu disederhanakan. Salah satu cara untuk menyederhanakan diagram kelas adalah dengan hanya menampilkan kelas konkret. Namun, tergantung pada jumlah asosiasi yang terhubung ke kelas abstrak — dan dengan demikian diwariskan ke kelas konkret — saran khusus ini dapat membuat diagram lebih sulit untuk dipahami.

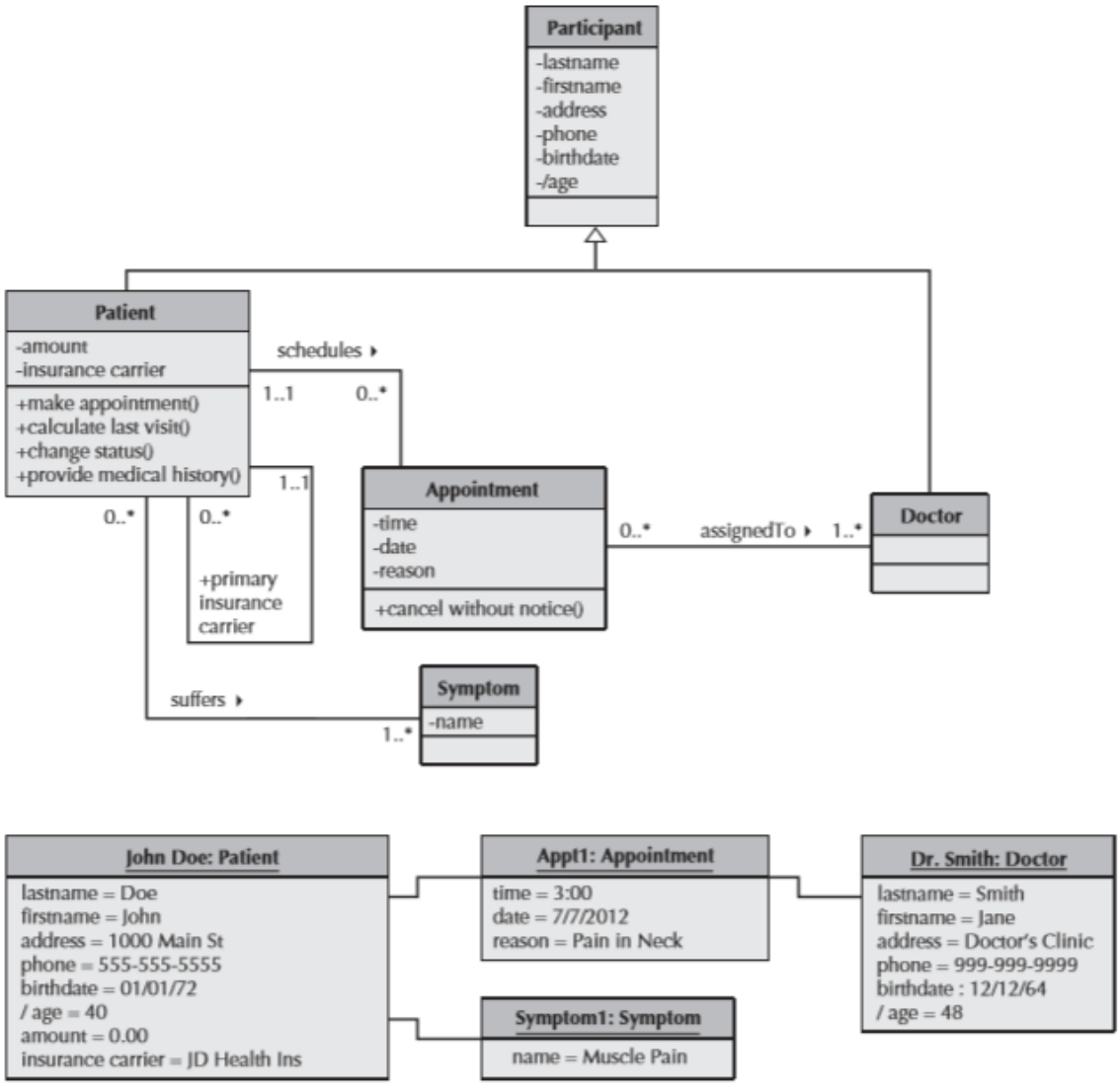
Cara kedua untuk menyederhanakan diagram kelas adalah melalui penggunaan mekanisme tampilan. Tampilan awalnya dikembangkan dengan sistem manajemen database relasional untuk menunjukkan hanya sebagian dari informasi yang terkandung dalam database. Dalam hal ini, tampilan akan menjadi bagian yang berguna dari diagram kelas, seperti tampilan Use-Case yang hanya menunjukkan kelas dan hubungan yang relevan dengan Use-Case tertentu. Pandangan kedua dapat menunjukkan hanya jenis hubungan tertentu: agregasi, asosiasi, atau generalisasi. Jenis tampilan ketiga adalah untuk membatasi informasi yang ditampilkan dengan setiap kelas, misalnya, hanya menampilkan nama kelas, nama dan atribut, atau nama dan operasi. Mekanisme tampilan ini dapat digabungkan untuk lebih menyederhanakan diagram.

Pendekatan ketiga untuk menyederhanakan diagram kelas adalah melalui penggunaan paket (yaitu, kelompok kelas logis). Untuk membuat diagram lebih mudah dibaca dan menjaga model pada tingkat kompleksitas yang wajar, kelas dapat dikelompokkan bersama ke dalam paket. Paket adalah konstruksi umum yang dapat diterapkan ke salah satu elemen dalam model UML. Dalam Bab 4, kami memperkenalkan ide paket untuk menyederhanakan diagram Use-Case. Dalam kasus diagram kelas, mudah untuk mengurutkan kelas ke dalam kelompok berdasarkan hubungan yang mereka bagikan.

Diagram Objek

Meskipun diagram kelas diperlukan untuk mendokumentasikan struktur kelas, tipe kedua dari diagram struktur statis, yang disebut diagram objek, dapat berguna dalam mengungkapkan informasi tambahan. Diagram objek pada dasarnya adalah instantiasi dari semua atau sebagian dari diagram kelas. Instansiasi berarti membuat instance kelas dengan sekumpulan nilai atribut yang sesuai.

Diagram objek bisa sangat berguna saat mencoba mengungkap detail kelas. Secara umum, lebih mudah untuk berpikir dalam kerangka objek konkret (contoh) daripada abstraksi objek (kelas). Misalnya pada Gambar 5-15, sebagian dari diagram kelas pada Gambar 5-7 telah disalin dan dipakai. Bagian atas gambar hanyalah salinan dari tampilan kecil diagram kelas secara keseluruhan. Bagian bawah adalah diagram objek yang menginstansiasi subset kelas tersebut. Dengan meninjau contoh aktual yang terlibat, John Doe, Appt1, Symptom1, dan Dr. Smith, kami dapat menemukan atribut, hubungan, dan/atau operasi tambahan yang relevan atau mungkin atribut, hubungan, dan/atau operasi yang salah tempat. Misalnya, janji temu memiliki atribut alasan. Setelah pemeriksaan lebih dekat, atribut alasan mungkin lebih baik dimodelkan sebagai asosiasi dengan kelas Gejala. Saat ini, kelas Gejala dikaitkan dengan kelas Pasien. Setelah meninjau diagram objek, ini tampaknya salah. Oleh karena itu, kita harus memodifikasi diagram kelas untuk mencerminkan pemahaman baru tentang masalah ini.



Gambar 5-15 Contoh Diagram Objek

5.7 MEMBUAT MODEL STRUKTUR MENGGUNAKAN KARTU CRC DAN DIAGRAM KELAS

Membuat model struktural adalah proses inkremental dan berulang di mana analis membuat potongan kasar model dan kemudian menyempurnakannya dari waktu ke waktu. Model struktural bisa menjadi sangat kompleks—bahkan, ada sistem yang memiliki ratusan kelas. Penting untuk diingat bahwa kartu CRC dan diagram kelas dapat digunakan untuk menggambarkan model struktural apa adanya dan yang akan datang dari sistem yang sedang berkembang, tetapi mereka paling sering digunakan untuk model yang akan dibuat. Ada banyak cara berbeda untuk mengidentifikasi sekumpulan calon objek dan membuat kartu CRC dan diagram kelas. Saat ini sebagian besar identifikasi objek dimulai dengan Use-Case yang diidentifikasi untuk masalah tersebut (lihat Bab 4). Pada bagian ini, kami menjelaskan proses use-case-driven yang dapat digunakan untuk membuat model struktural dari domain masalah.

Front:

Class Name: Apartment Owner	**ID:** 1	**Type:** Concrete, Domain
Description: An apartment owner who has apartments for rent		**Associated Use Cases:** 2

Responsibilities	**Collaborators**
Add apartment	Apartment
Delete apartment	Apartment

Back:

Attributes:

Name (string)	
Address (address)	
Phone number (PhoneNumber)	
Email (EmailAddress)	

Relationships:

Generalization (a-kind-of):

Aggregation (has-parts):

Other Associations: Apartment

Gambar 5-16 Kartu CRC Pemilik Apartemen Perumahan Kampus

Kita bisa mulai membuat model struktural dengan diagram kelas alih-alih kartu CRC. Namun, karena sifat teknologi rendah dan kemudahan skenario Use-Case role-playing dengan kartu CRC, kami lebih memilih untuk membuat kartu CRC terlebih dahulu dan kemudian mentransfer informasi dari kartu CRC ke dalam diagram kelas nanti. Akibatnya, langkah pertama dari proses yang kami rekomendasikan adalah membuat kartu CRC. Melakukan analisis tekstual pada deskripsi use-case melakukan ini. Jika Anda ingat, aliran normal peristiwa, subalur, dan aliran alternatif/luar biasa dari deskripsi Use-Case ditulis dalam bentuk khusus yang disebut Subjek– Kata Kerja–Objek Langsung–Preposisi–Objek tidak langsung (SVDPI). Dengan menulis kejadian Use-Case dalam formulir ini, akan lebih mudah untuk menggunakan pedoman analisis tekstual pada Gambar 5-1 untuk mengidentifikasi objek. Meninjau aktor utama, pemegang kepentingan dan kepentingan, dan deskripsi singkat dari setiap Use-Case memungkinkan objek kandidat tambahan untuk diidentifikasi. Sangat berguna untuk kembali dan meninjau persyaratan asli untuk mencari informasi yang tidak disertakan dalam teks Use-Case. Catat semua informasi yang ditemukan untuk setiap objek kandidat pada kartu CRC.

Langkah kedua adalah meninjau kartu CRC untuk menentukan apakah objek kandidat tambahan, atribut, operasi, dan hubungan hilang. Sehubungan dengan tinjauan ini, menggunakan pendekatan brainstorming dan daftar objek umum yang dijelaskan sebelumnya dapat membantu tim dalam mengidentifikasi kelas, atribut, operasi, dan hubungan yang hilang. Misalnya, tim dapat memulai sesi curah pendapat dengan serangkaian pertanyaan seperti:

- Apa saja hal-hal nyata yang terkait dengan masalah?
- Apa peran yang dimainkan oleh orang-orang dalam domain masalah?
- Insiden dan interaksi apa yang terjadi dalam domain masalah?

Seperti yang dapat Anda lihat dengan mudah, dengan memulai dengan deskripsi Use-Case, banyak dari pertanyaan ini sudah memiliki sebagian jawaban. Misalnya, aktor dan pemegang kepentingan utama adalah peran yang dimainkan oleh orang-orang dalam domain masalah. Namun, dimungkinkan untuk mengungkap peran tambahan yang tidak terpikirkan sebelumnya. Ini jelas akan menyebabkan deskripsi use-case, dan mungkin diagram use-case, dimodifikasi dan mungkin diperluas. Seperti pada langkah sebelumnya, pastikan untuk mencatat semua informasi yang tidak terungkap ke dalam kartu CRC. Ini termasuk modifikasi apa pun yang ditemukan untuk objek kandidat yang diidentifikasi sebelumnya dan informasi apa pun mengenai objek kandidat baru yang diidentifikasi.

Langkah ketiga adalah memainkan peran setiap skenario Use-Case menggunakan kartu CRC. Setiap kartu CRC harus diberikan kepada individu yang akan melakukan operasi untuk kelas pada kartu CRC. Saat para pemain memainkan peran mereka, sistem cenderung rusak. Ketika ini terjadi, objek, atribut, operasi, atau hubungan tambahan akan diidentifikasi. Sekali lagi, seperti pada langkah sebelumnya, setiap kali informasi baru ditemukan, kartu CRC baru dibuat atau modifikasi kartu CRC yang ada dibuat.

Langkah keempat adalah membuat diagram kelas berdasarkan kartu CRC. Informasi yang terdapat pada kartu CRC ditransfer ke diagram kelas. Tanggung jawab dialihkan sebagai operasi; atribut digambar sebagai atribut; dan hubungan digambarkan sebagai hubungan generalisasi, agregasi, atau asosiasi. Namun, diagram kelas juga mengharuskan visibilitas atribut dan operasi diketahui. Sebagai aturan umum, atribut bersifat pribadi dan operasi bersifat publik. Oleh karena itu, kecuali jika analis memiliki alasan yang baik untuk mengubah visibilitas default dari properti ini, maka default tersebut harus diterima. Akhirnya, analis harus memeriksa model untuk peluang tambahan untuk menggunakan hubungan agregasi atau

generalisasi. Jenis hubungan ini dapat menyederhanakan deskripsi kelas individu. Seperti pada langkah sebelumnya, semua perubahan harus dicatat pada kartu CRC.

Langkah kelima adalah meninjau model struktural untuk kelas, atribut, operasi, dan hubungan yang hilang dan/atau tidak diperlukan. Sampai langkah ini, fokus prosesnya adalah pada penambahan informasi ke model yang sedang berkembang. Pada titik ini, fokus mulai beralih dari sekadar menambahkan informasi menjadi juga menantang alasan untuk memasukkan informasi yang terkandung dalam model. Salah satu pendekatan yang sangat berguna di sini adalah bermain sebagai advokat iblis, di mana seorang anggota tim, hanya demi rasa sakit di leher, menantang alasan untuk memasukkan semua aspek model.

Langkah keenam adalah memasukkan pola yang berguna ke dalam model struktural yang berkembang. Sebuah pola yang berguna adalah salah satu yang akan memungkinkan analis untuk lebih lengkap menggambarkan domain yang mendasari masalah yang sedang diselidiki. Melihat koleksi pola yang tersedia (Gambar 5-5) dan membandingkan kelas yang terdapat dalam pola dengan yang ada di diagram kelas yang berkembang memungkinkan hal ini. Setelah mengidentifikasi pola yang berguna, analis menggabungkan pola yang diidentifikasi ke dalam diagram kelas dan memodifikasi kartu CRC yang terpengaruh. Ini termasuk menambahkan dan menghapus kelas, atribut, operasi, dan/atau hubungan.

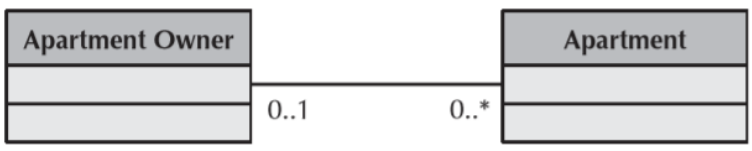
Langkah ketujuh dan terakhir adalah memvalidasi model struktural, termasuk kartu CRC dan diagram kelas. Kami membahas konten ini di bagian berikutnya dari bab ini dan di Bab 7.

Contoh Perumahan Kampus

Pada bab sebelumnya, kami mengidentifikasi serangkaian Use-Case (Tambah Apartemen, Hapus Apartemen, dan Cari Unit Sewa yang Tersedia) untuk layanan perumahan kampus yang membantu siswa menemukan apartemen. Dengan meninjau Use-Case, kita dapat dengan mudah menentukan bahwa layanan perumahan kampus harus melacak informasi untuk setiap apartemen yang tersedia dan pemiliknya. Informasi yang akan ditangkap untuk setiap apartemen adalah lokasi apartemen, jumlah kamar tidur di apartemen, sewa bulanan, dan seberapa jauh apartemen dari kampus. Mengenai pemilik apartemen, kita perlu menangkap informasi kontak pemilik (misalnya, nama, alamat, nomor telepon, alamat email). Karena siswa hanyalah pengguna sistem, tidak perlu menangkap informasi apa pun tentang mereka; yaitu, dalam hal ini, siswa hanyalah aktor. Akhirnya, berkaitan dengan hubungan antar kelas, ada hubungan asosiasi tunggal, opsional, satu ke banyak yang menghubungkan dua kelas bersama-sama. Kartu CRC Pemilik Apartemen digambarkan pada Gambar 5-16, dan diagram kelas untuk situasi ini ditunjukkan pada Gambar 5-17.

Contoh Perpustakaan

Seperti contoh Perumahan Kampus, langkah pertama adalah membuat kartu CRC yang mewakili kelas-kelas dalam model struktural. Pada bab sebelumnya, kami menggunakan contoh Sistem Manajemen Koleksi Buku Perpustakaan untuk menggambarkan proses pembuatan model fungsional (use-case dan diagram aktivitas dan deskripsi use-case). Dalam bab ini, kita mengikuti contoh familiar yang sama. Karena kami mengikuti pendekatan berbasis Use-Case untuk pengembangan sistem berorientasi objek, pertama-tama kami meninjau peristiwa yang dijelaskan dalam deskripsi Use-Case (lihat Gambar 5-18).



Gambar 5-17 Diagram Kelas Perumahan Kampus

Selanjutnya, kami melakukan analisis tekstual pada peristiwa dengan menerapkan aturan analisis tekstual yang dijelaskan pada Gambar 5-1. Dalam hal ini, kita dapat dengan cepat mengidentifikasi kebutuhan untuk memasukkan kelas untuk Peminjam, Buku, Pustakawan, Meja Check Out, ID Card, Peminjam Mahasiswa, Peminjam Fakultas/Staf, Peminjam Tamu, Database Panitera, Database Personil, Database Tamu Perpustakaan, Buku Terlambat , Denda, Permintaan Buku. Kami juga dapat dengan mudah mengidentifikasi operasi untuk "memeriksa validitas" permintaan buku, untuk "memeriksa" buku, dan untuk "menolak" permintaan buku. Lebih lanjut, peristiwa tersebut menunjukkan hubungan "membawa" antara Peminjam dan Buku dan hubungan "menyediakan" antara Peminjam dan Pustakawan. Langkah ini juga menyarankan bahwa kita harus meninjau bagian ikhtisar dari deskripsi use-case (lihat Gambar 5-19). Dalam hal ini, satu-satunya informasi tambahan yang diperoleh dari deskripsi Use-Case adalah kemungkinan dimasukkannya kelas untuk Kantor Personalia dan Kantor Panitera. Proses yang sama ini juga akan diselesaikan untuk Use-Case yang tersisa yang terdapat dalam model fungsional: Proses Buku yang Terlambat, Memelihara Koleksi Buku, Mencari Koleksi, dan Mengembalikan Buku (lihat Gambar 4-6). Karena kita tidak membahas Use-Case ini di bab sebelumnya, kita akan meninjau deskripsi masalah sebagai dasar untuk memulai langkah berikutnya (lihat Gambar 5-20).

Alur Acara Normal: 1. Peminjam membawa buku ke Pustakawan di meja check out. 2. Peminjam memberikan Kartu Tanda Penduduk kepada Pustakawan. 3. Pustakawan memeriksa keabsahan KTP. Jika Peminjam adalah Peminjam Pelajar, Validasi KTP terhadap Database Panitera. Jika Peminjam adalah Peminjam Fakultas/Staf, Validasi KTP terhadap Database Kepegawaian. Jika Peminjam adalah Peminjam Tamu, Validasi KTP terhadap Database Tamu Perpustakaan. 4. Pustakawan memeriksa apakah Peminjam memiliki buku dan/atau denda yang terlambat. 5. Peminjam memeriksa buku.
SubAliran:
Aliran Alternatif/Luar Biasa: 4a. ID Card tidak valid, permintaan buku ditolak. 5a. Peminjam memiliki denda buku yang terlambat, atau keduanya, permintaan buku ditolak.

Gambar 5-18 Deskripsi Alur Use Case Pinjam Buku (Gambar 4-21)

Nama Use Case : Borrow Book	ID : 2	Level Kesulitan : High
Primary Actor: Borrower	Use-Case Type: Detail, Real	
Stakeholders and Interests: Peminjam—ingin memeriksa buku Pustakawan—ingin memastikan peminjam hanya mendapatkan buku yang layak		
Deskripsi Singkat: Use case ini menjelaskan bagaimana buku-buku dikeluarkan dari perpustakaan..		

Pemicu:	Peminjam membawa buku untuk memeriksa meja..
Type :	External

Relationships:	
	Association:
	Sales Rep Include:
	Extend:
Generalization:	

Gambar 5-19 Deskripsi Ikhtisar Use Case Pinjam Buku (Gambar 4-20)

Persyaratan fungsional untuk sistem sirkulasi perpustakaan universitas otomatis mencakup kebutuhan untuk mendukung kegiatan pencarian, peminjaman, dan pemeliharaan buku. Sistem harus mendukung pencarian berdasarkan judul, penulis, kata utama, dan ISBN. Pencarian database koleksi perpustakaan harus tersedia di terminal di perpustakaan dan tersedia untuk calon peminjam melalui World Wide Web. Jika buku yang diminati saat ini sedang diperiksa, peminjam yang sah harus diizinkan untuk meminta buku itu dikembalikan. Setelah buku diperiksa kembali, peminjam yang meminta buku harus diberitahu tentang ketersediaan buku.

Kegiatan peminjaman dibangun di sekitar memeriksa buku dan mengembalikan buku oleh peminjam. Ada tiga jenis peminjam: mahasiswa, dosen dan staf, dan tamu. Terlepas dari jenis peminjam, peminjam harus memiliki KTP yang masih berlaku. Jika peminjam adalah mahasiswa, sistem akan memeriksa dengan database mahasiswa pendaftar memvalidasi kartu ID. Jika peminjam adalah staf pengajar atau staf, sistem memeriksa dengan database karyawan kantor personalia memvalidasi kartu identitas. Jika peminjam adalah tamu, kartu ID diperiksa terhadap database peminjam perpustakaan itu sendiri. Jika KTP masih berlaku, sistem juga harus memeriksa untuk menentukan apakah peminjam memiliki buku yang terlambat atau denda yang belum dibayar. Jika KTP tidak berlaku, peminjam memiliki tunggakan buku, atau peminjam memiliki denda yang belum dibayar, sistem harus menolak permintaan peminjam untuk memeriksa buku; jika tidak, permintaan peminjam harus dihormati. Jika sebuah buku di-check out, sistem harus memperbarui database koleksi perpustakaan untuk mencerminkan status baru buku tersebut.

Kegiatan pemeliharaan buku berkaitan dengan penambahan dan penghapusan buku dari koleksi buku perpustakaan. Ini membutuhkan manajer perpustakaan untuk menambah dan menghapus buku secara logis dan fisik. Buku yang dibeli oleh perpustakaan atau buku yang dikembalikan dalam keadaan rusak biasanya menyebabkan aktivitas ini. Jika sebuah buku dipastikan rusak saat dikembalikan dan perlu dikeluarkan dari koleksi, peminjam terakhir akan dikenakan denda. Namun, jika buku dapat diperbaiki, tergantung pada biaya perbaikan, peminjam mungkin tidak akan dikenakan denda. Akhirnya, setiap hari Senin, perpustakaan mengirimkan e-mail pengingat kepada peminjam yang memiliki buku terlambat. Jika sebuah buku terlambat lebih dari dua minggu, peminjam dinilai denda. Tergantung pada berapa lama buku tersebut jatuh tempo, peminjam dapat dikenakan denda tambahan setiap hari Senin.

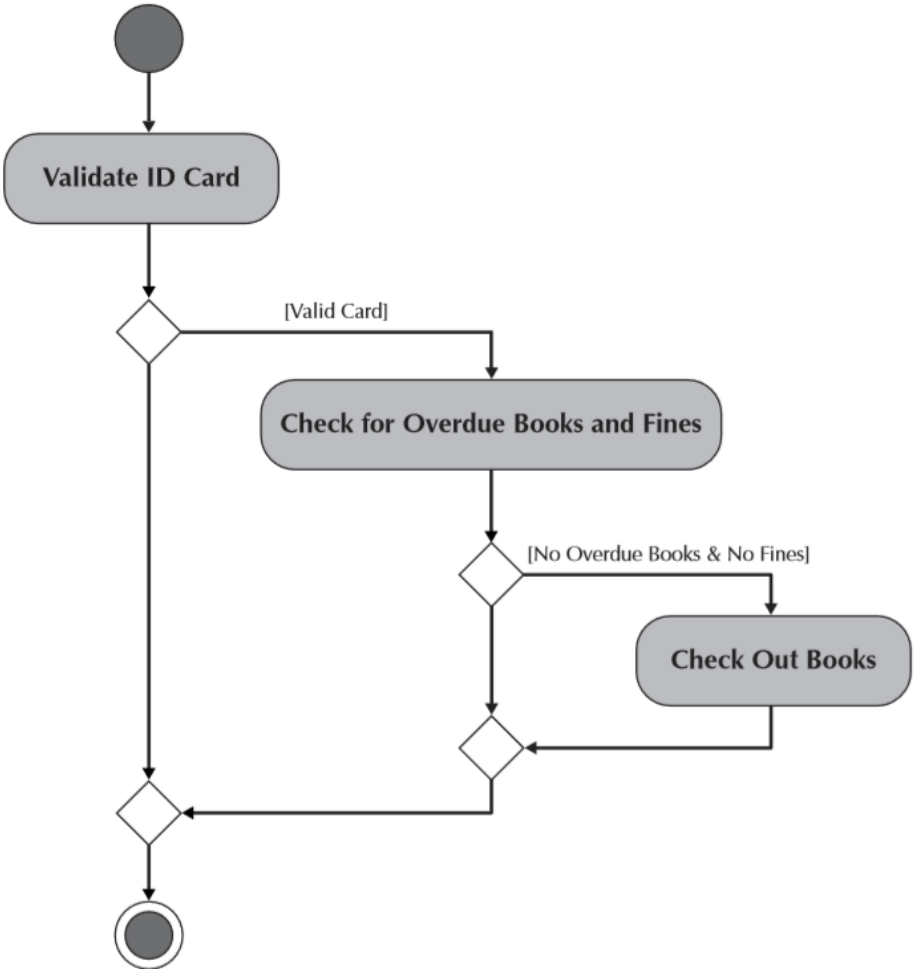
Gambar 5-20 Gambaran Umum Deskripsi Sistem Manajemen Koleksi Buku Perpustakaan

Langkah kedua adalah meninjau kartu CRC untuk menentukan apakah ada informasi yang hilang. Dalam kasus sistem perpustakaan, karena kami hanya menggunakan deskripsi Use-Case Pinjam Buku, beberapa informasi jelas hilang. Dengan meninjau Gambar 5-20, kita melihat bahwa kita perlu menyertakan kemampuan untuk mencari koleksi buku berdasarkan judul, pengarang, kata utama, dan ISBN. Ini jelas menyiratkan kelas Koleksi Buku dengan empat operasi pencarian yang berbeda: Pencarian Berdasarkan Judul, Pencarian Berdasarkan Penulis, Pencarian Berdasarkan Kata Utama, dan Pencarian Berdasarkan ISBN. Menariknya, deskripsi juga menyiratkan satu set subclass atau status untuk kelas Buku: Check Out, Overdue, Requested, Available, dan Damaged. Kami akan kembali ke masalah negara bagian versus subkelas di bab berikutnya. Deskripsi menyiratkan banyak operasi tambahan, termasuk Mengembalikan Buku, Meminta Buku, Menambah Buku, Menghapus Buku, Memperbaiki Buku, Meminjam Denda, dan Mengirim Email Pengingat.

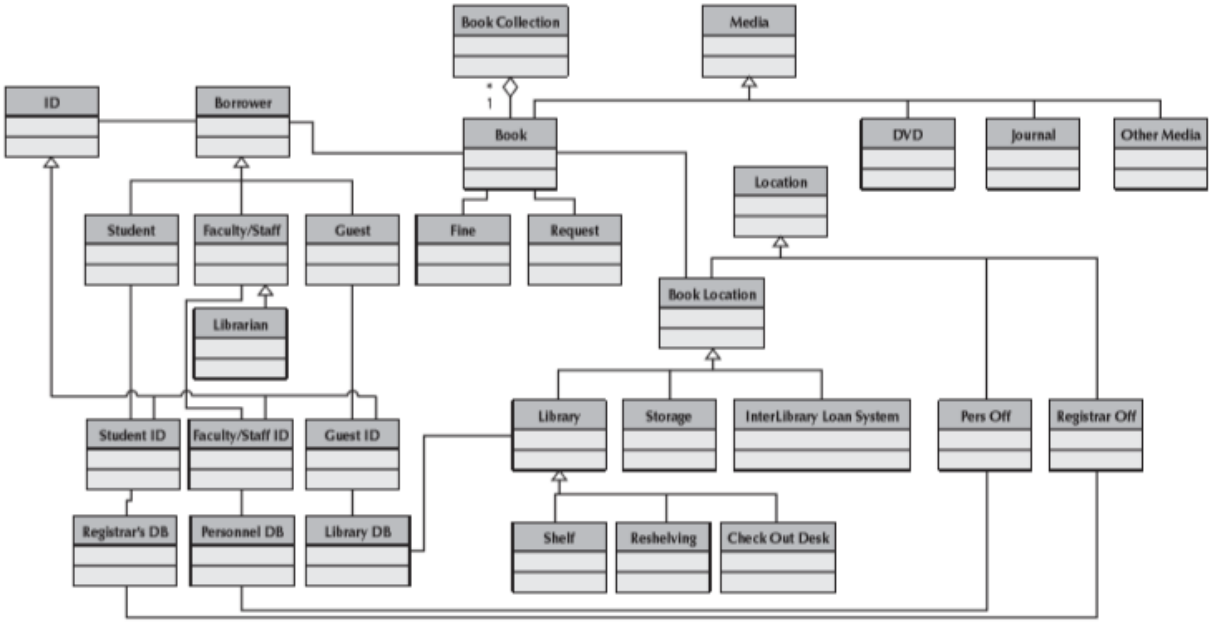
Selanjutnya, kita harus menggunakan pengalaman perpustakaan kita sendiri untuk melakukan brainstorming potensi kelas, atribut, operasi, dan hubungan tambahan yang dapat berguna untuk disertakan dalam Sistem Manajemen Koleksi Buku Perpustakaan. Di perpustakaan kami, ada juga kebutuhan untuk Mengambil Buku Dari Penyimpanan, Memindahkan Buku ke Penyimpanan, Meminta Buku dari Sistem Pinjaman Antar Perpustakaan, Mengembalikan Buku ke Sistem Pinjaman Antar Perpustakaan, dan Menangani E-Buku. Anda juga dapat memasukkan kelas untuk Jurnal, DVD, dan media lainnya. Seperti yang Anda lihat, banyak kelas, atribut, operasi, dan hubungan dapat diidentifikasi.

Langkah ketiga, bermain peran kartu CRC, mengharuskan kita untuk menerapkan tiga langkah bermain peran yang dijelaskan sebelumnya:

- Tinjau Use-Case
- Identifikasi Aktor dan Objek yang Relevan
- Skenario Bermain Peran



Gambar 5-21 Activity Diagram Use Case Pinjam Buku (Gambar 4-12)

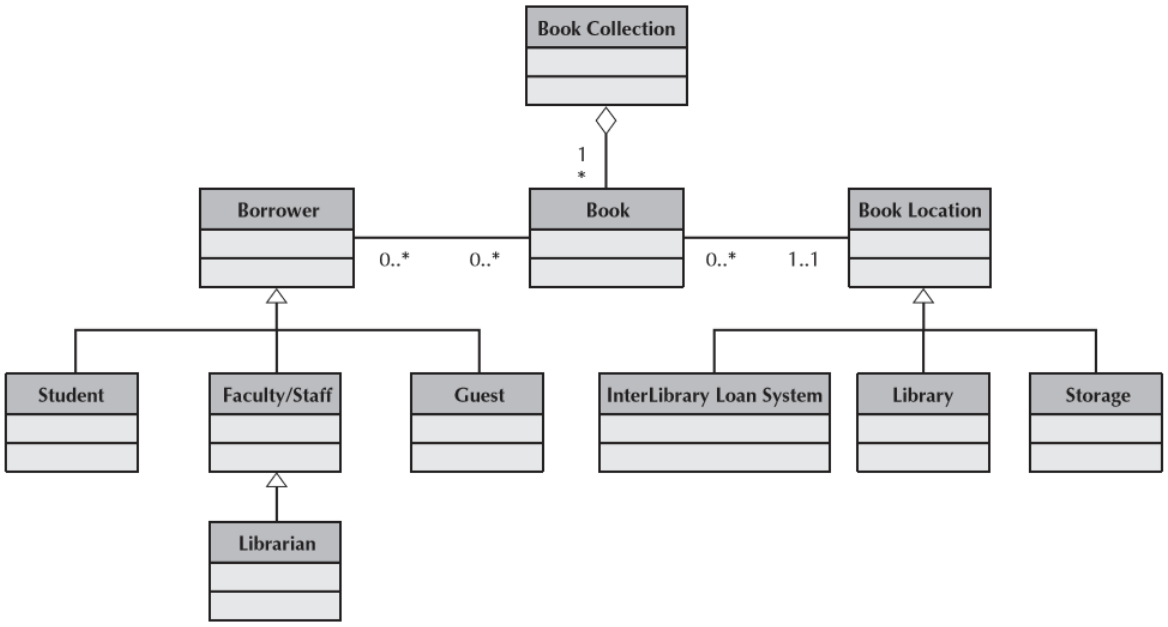


Gambar 5-22 Diagram First-Cut Class untuk Sistem Koleksi Buku Perpustakaan Book

Untuk tujuan kami, kami akan menggunakan Use-Case Pinjam Buku untuk menunjukkan. Aktor-aktor terkait antara lain Peminjam Mahasiswa, Peminjam Fakultas/Staf, Peminjam Tamu, Pustakawan, Bagian Personalia, dan Kepaniteraan. Ini dapat dengan mudah

diperoleh dari bagian ikhtisar deskripsi use-case (lihat Gambar 5-19) dan diagram use-case (lihat Gambar 4-6). Obyek yang relevan tampaknya termasuk Buku, Peminjam, dan KTP. Akhirnya, untuk memainkan skenario, kita perlu menetapkan peran ke anggota tim yang berbeda dan mencoba untuk melakukan setiap jalur melalui peristiwa Use-Case (lihat Gambar 5-18). Berdasarkan Kejadian dari use case dan diagram aktivitas use case (lihat Gambar 5-21), kita dapat dengan cepat mengidentifikasi sembilan skenario, tiga untuk setiap jenis Peminjam (Mahasiswa, Fakultas/Staf dan Tamu): Valid ID dan No Overdue Buku & Tanpa Denda, Hanya ID yang Valid, dan tanpa ID yang Valid. Saat memainkan skenario ini, satu pertanyaan muncul: Apa yang terjadi pada buku yang diminta saat permintaan ditolak? Berdasarkan model fungsional dan struktural saat ini, buku-buku dibiarkan duduk di meja check out. Itu sepertinya tidak benar. Kenyataannya, buku-buku itu dirapikan kembali. Sebenarnya, pengertian menata ulang buku juga relevan dengan saat buku dicek kembali atau setelah buku diperbaiki. Selain itu, ide penambahan buku ke dalam koleksi juga harus mencakup pengoperasian rak buku. Seperti yang akan segera Anda lihat, membangun model struktural juga akan membantu mengungkap perilaku yang dihilangkan saat membangun model fungsional. Ingat, pengembangan sistem berorientasi objek tidak hanya didorong oleh Use-Case tetapi juga bersifat inkremental dan berulang.

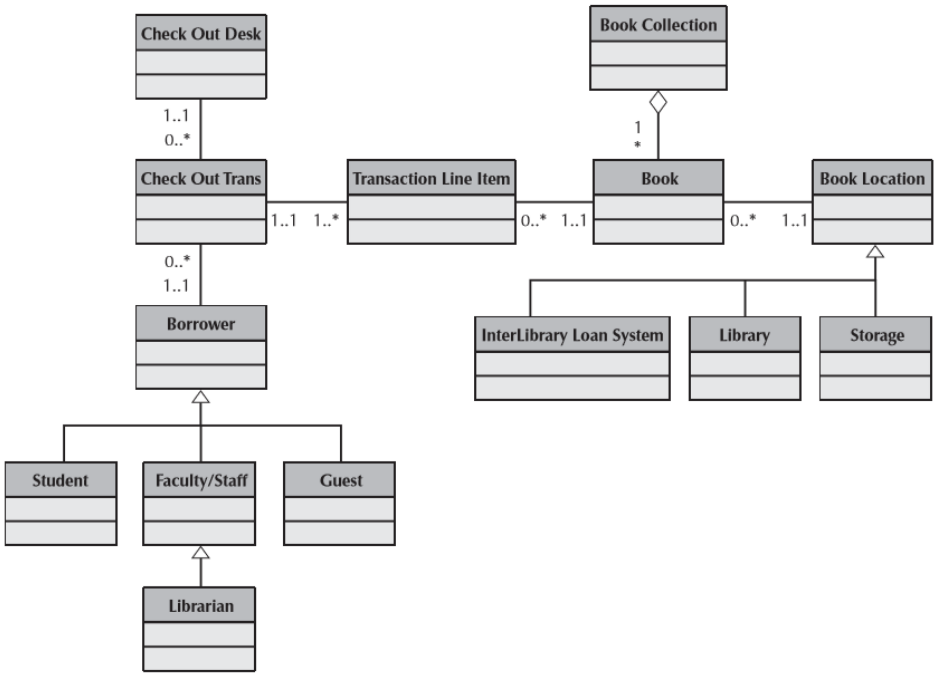
Langkah keempat adalah menyatukan semuanya dan menggambar diagram kelas. Gambar 5-22 menunjukkan potongan pertama dalam menggambar diagram kelas untuk Sistem Manajemen Koleksi Buku Perpustakaan. Kelas yang diidentifikasi pada langkah sebelumnya telah dihubungkan dengan kelas lain melalui hubungan asosiasi, agregasi, dan generalisasi. Untuk tujuan kesederhanaan, kami hanya menampilkan kelas dan hubungannya; bukan atribut mereka, operasi, atau bahkan multiplisitas pada hubungan asosiasi.



Gambar 5-23 Second-Cut Class Diagram untuk Sistem Koleksi Buku Perpustakaan

Langkah kelima adalah dengan hati-hati meninjau apa yang telah dibuat. Anda tidak hanya harus mencari kelas, atribut, operasi, dan/atau hubungan yang hilang, tetapi Anda juga harus menantang setiap aspek dari model saat ini. Secara khusus, apakah ada kelas, atribut, operasi, dan/atau hubungan yang harus dihapus dari model? Jika demikian, mungkin ada kelas pada diagram yang seharusnya dimodelkan sebagai atribut. Sebagai contoh, Student, Fac/Staff, dan Guest ID harus memiliki atribut dengan class masing-masing. Selain itu, karena ini adalah sistem pengelolaan koleksi buku, penyertaan media lain tampaknya tidak tepat. Terakhir, Biro Kepegawaian dan Kepaniteraan sebenarnya hanya aktor dalam sistem, bukan objek.

Berdasarkan semua penghapusan ini, versi baru dari diagram kelas digambar (lihat Gambar 5-23). Diagram ini jauh lebih sederhana dan lebih mudah dipahami.



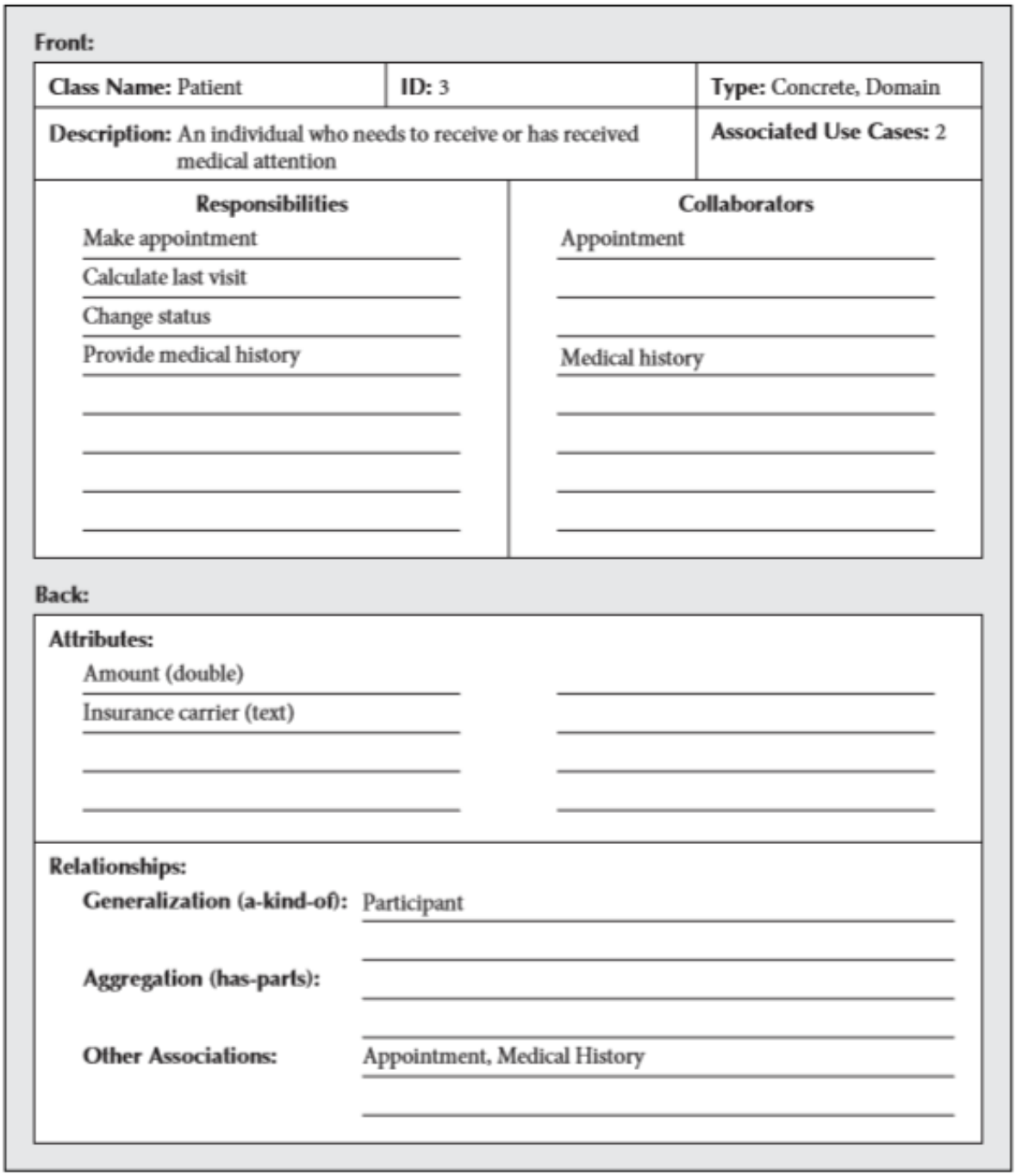
Gambar 5-24 Diagram Kelas dengan Pola Terpadu untuk Sistem Pengumpulan Buku Perpustakaan

Langkah keenam, menggabungkan pola-pola yang bermanfaat, memungkinkan kita memanfaatkan pengetahuan yang dikembangkan di tempat lain. Dalam hal ini, pola yang digunakan dalam masalah perpustakaan terlalu banyak memasukkan ide-ide yang tidak relevan dengan masalah saat ini. Namun, dengan melihat kembali ke Gambar 5-3, kita melihat bahwa salah satu pola asli (pola Tempat, Transaksi, Peserta, Item Baris Transaksi, dan Item—lihat kiri atas gambar) adalah relevan. Kami memasukkan pola tersebut ke dalam diagram kelas dengan mengganti Tempat dengan Meja Check Out, Peserta berdasarkan Peminjam, Transaksi dengan Trans Check Out, dan Item per Buku (Gambar 5-24). Secara teknis, masing-masing penggantian ini hanyalah sebuah pola yang disesuaikan dengan masalah yang dihadapi. Kami juga kemudian menambahkan kelas Item Baris Transaksi yang telah kami lewatkan dalam model struktural asli.

Langkah ketujuh adalah meninjau keadaan model struktural saat ini. Tak perlu dikatakan, versi kartu CRC dan versi diagram kelas tidak lagi sesuai satu sama lain. Kita kembali ke langkah ini di bagian berikutnya dari bab ini.

5.8 VERIFIKASI DAN VALIDASI MODEL STRUKTUR

Sebelum kita beralih ke pembuatan model perilaku (lihat Bab 6) dari domain masalah, kita perlu memverifikasi dan memvalidasi model struktural. Pada bab sebelumnya, kami memperkenalkan gagasan walkthrough sebagai cara untuk memverifikasi dan memvalidasi proses bisnis dan model fungsional. Dalam bab ini, kami menggabungkan penelusuran dengan kekuatan bermain peran sebagai cara untuk lebih lengkap memverifikasi dan memvalidasi model struktural yang akan mendasari proses bisnis dan model fungsional. Faktanya, semua pendekatan identifikasi objek yang dijelaskan dalam bab ini dapat dilihat sebagai cara untuk menguji kesetiaan model struktural. Karena kami telah memperkenalkan ide bermain peran kartu CRC dan identifikasi objek, di bagian ini kami fokus pada melakukan penelusuran.



Front:

Class Name: Patient	**ID:** 3	**Type:** Concrete, Domain
Description: An individual who needs to receive or has received medical attention		**Associated Use Cases:** 2

Responsibilities	Collaborators
Make appointment	Appointment
Calculate last visit	
Change status	
Provide medical history	Medical history

Back:

Attributes:

Amount (double)	
Insurance carrier (text)	

Relationships:

Generalization (a-kind-of): Participant

Aggregation (has-parts):

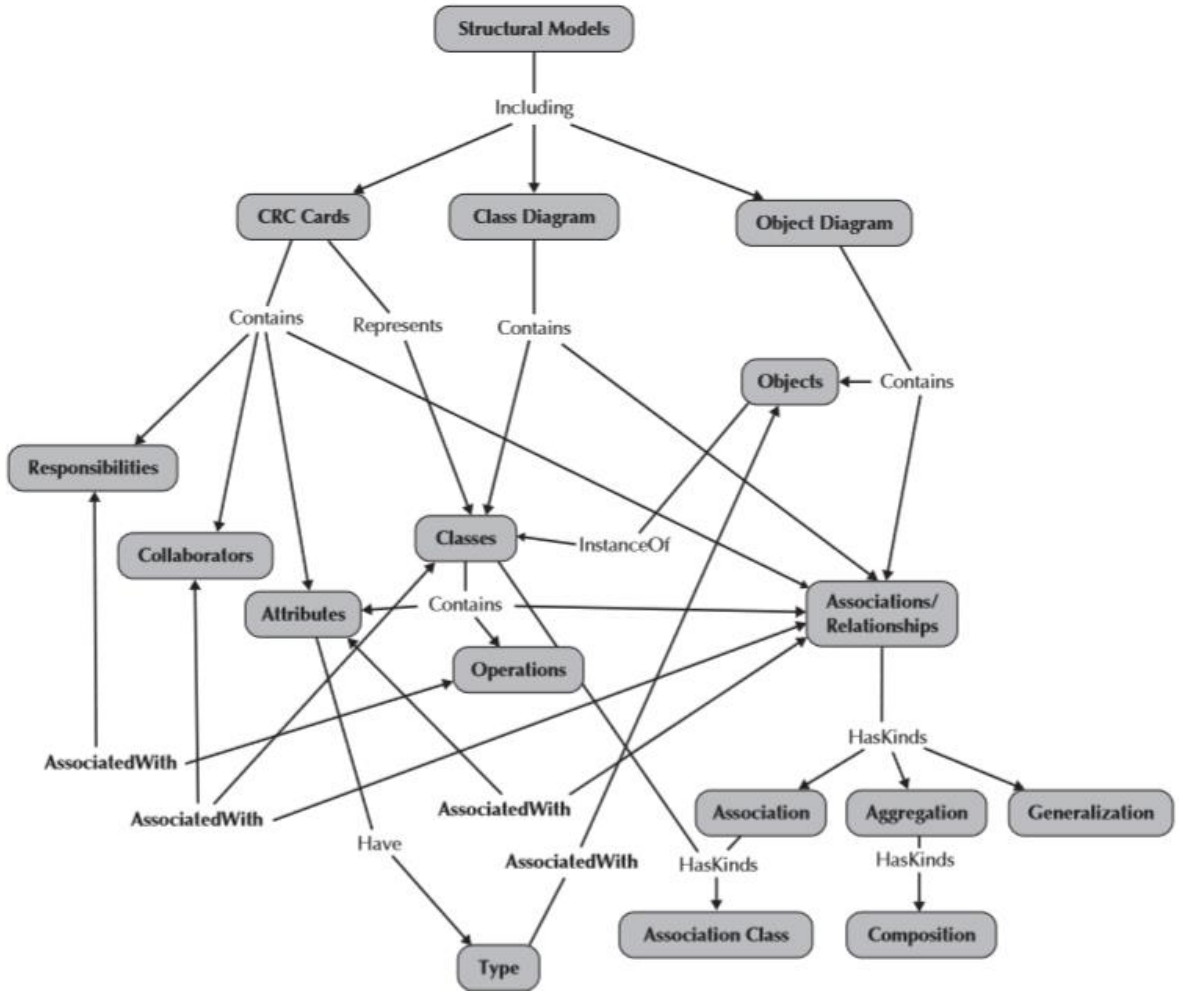
Other Associations: Appointment, Medical History

Gambar 5-25 Kartu CRC Pasien

Dalam hal ini, verifikasi dan validasi model struktural dilakukan selama pertemuan tinjauan formal menggunakan pendekatan walkthrough di mana seorang analis menyajikan model kepada tim pengembang dan pengguna. Analis berjalan melalui model, menjelaskan setiap bagian dari model dan semua alasan di balik keputusan untuk memasukkan masing-masing kelas dalam model struktural. Penjelasan ini mencakup pembenaran untuk atribut, operasi, dan hubungan yang terkait dengan kelas. Setiap kelas harus ditautkan kembali ke setidaknya satu Use-Case; jika tidak, tujuan memasukkan kelas dalam model struktural tidak akan dipahami. Juga termasuk orang-orang di luar tim pengembangan yang menghasilkan model dapat membawa perspektif baru ke model dan mengungkap objek yang hilang.

Sebelumnya, kami menyarankan tiga representasi yang dapat digunakan untuk pemodelan struktural: kartu CRC, diagram kelas, dan diagram objek. Karena diagram objek

hanyalah instantiasi dari beberapa bagian diagram kelas, kami membatasi diskusi kami pada kartu CRC dan diagram kelas. Mirip dengan bagaimana kami memverifikasi dan memvalidasi proses bisnis dan model fungsional di bab terakhir, kami menyediakan seperangkat aturan yang akan menguji konsistensi dalam model struktural. Sebagai contoh tujuan, kami menggunakan masalah janji yang dijelaskan dalam Bab 4 dan dalam bab ini. Contoh kartu CRC untuk kelas pasien lama ditunjukkan pada Gambar 5-6, dan diagram kelas terkait digambarkan pada Gambar 5-7.



Gambar 5-26 Hubungan Antar Model Struktural

Pertama, setiap kartu CRC harus dikaitkan dengan kelas pada diagram kelas, dan sebaliknya. Pada contoh janji temu, kelas Pasien Lama yang diwakili oleh kartu CRC tampaknya tidak disertakan pada diagram kelas. Namun, ada kelas Pasien pada diagram kelas (lihat Gambar 5-6 dan 5-7). Kartu CRC Pasien Lama kemungkinan besar harus diubah menjadi Pasien saja.

Kedua, tanggung jawab yang tercantum di bagian depan kartu CRC harus dimasukkan sebagai operasi di kelas pada diagram kelas, dan sebaliknya. Tanggung jawab membuat janji temu pada kartu CRC Pasien baru juga muncul sebagai operasi buat janji () di kelas Pasien pada diagram kelas. Setiap tanggung jawab dan operasi harus diperiksa.

Ketiga, kolaborator di bagian depan kartu CRC menyiratkan beberapa jenis hubungan di bagian belakang kartu CRC dan beberapa jenis asosiasi yang terhubung ke kelas terkait pada diagram kelas. Kolaborator janji temu di bagian depan kartu CRC juga muncul sebagai

asosiasi lain di bagian belakang kartu CRC dan sebagai asosiasi pada diagram kelas yang menghubungkan kelas Pasien dengan kelas Janji.

Keempat, atribut yang tertera pada bagian belakang kartu CRC harus dimasukkan sebagai atribut dalam suatu kelas pada diagram kelas, dan sebaliknya. Misalnya, atribut jumlah pada kartu CRC Pasien baru termasuk dalam daftar atribut kelas Pasien pada diagram kelas.

Kelima, tipe objek dari atribut-atribut yang tertera di bagian belakang kartu CRC dan dengan atribut-atribut dalam daftar atribut kelas pada diagram kelas menyiratkan asosiasi dari kelas ke kelas dari tipe objek. Misalnya, secara teknis, atribut jumlah menyiratkan asosiasi dengan tipe ganda. Namun, tipe sederhana seperti int dan double tidak pernah ditampilkan pada diagram kelas. Selanjutnya, tergantung pada domain masalah, tipe objek seperti Orang, Alamat, atau Tanggal mungkin juga tidak ditampilkan secara eksplisit. Namun, jika kita tahu bahwa pesan sedang dikirim ke instance dari tipe objek tersebut, kita mungkin harus menyertakan asosiasi tersirat ini sebagai hubungan.

Keenam, hubungan yang terdapat di bagian belakang kartu CRC harus digambarkan dengan menggunakan notasi yang sesuai pada diagram kelas. Misalnya pada Gambar 5-6, instance dari kelas Pasien adalah jenis Orang, memiliki instance dari kelas Riwayat Medis sebagai bagian darinya, dan memiliki asosiasi dengan instance dari kelas Appointment. Dengan demikian, asosiasi dari kelas Pasien ke kelas Person harus menunjukkan bahwa kelas Person adalah generalisasi dari subkelasnya, termasuk kelas Pasien; asosiasi dari kelas Pasien ke kelas Riwayat Medis harus dalam bentuk asosiasi agregasi (berlian putih); dan hubungan antara instance kelas Pasien dan instance kelas Appointment harus berupa asosiasi sederhana. Namun, ketika kita meninjau diagram kelas pada Gambar 5-7, bukan ini yang kita temukan. Jika Anda ingat, kami memasukkan dalam diagram kelas pola transaksi yang digambarkan pada Gambar 5-4. Ketika kami melakukan ini, banyak perubahan yang dibuat pada kelas-kelas yang terdapat dalam diagram kelas. Semua perubahan ini seharusnya mengalir kembali melalui semua kartu CRC. Dalam hal ini, kartu CRC untuk kelas Pasien harus menunjukkan bahwa Pasien adalah sejenis Peserta (bukan Orang) dan bahwa hubungan dari Pasien ke Riwayat Medis harus berupa asosiasi sederhana (lihat Gambar 5-25).

Ketujuh, kelas asosiasi, seperti kelas Treatment pada Gambar 5-7, harus dibuat hanya jika memang ada beberapa karakteristik unik (atribut, operasi, atau hubungan) tentang perpotongan kelas penghubung. Jika tidak ada karakteristik unik, maka kelas asosiasi harus dihapus dan hanya asosiasi antara dua kelas penghubung yang akan ditampilkan.

Akhirnya, seperti dalam model fungsional, aturan representasi tertentu harus ditegakkan. Misalnya, sebuah kelas tidak bisa menjadi subkelas dari dirinya sendiri. Kartu CRC Pasien tidak dapat mencantumkan Pasien dengan hubungan generalisasi di bagian belakang kartu CRC, juga tidak dapat ditarik hubungan generalisasi dari kelas Pasien ke dirinya sendiri. Sekali lagi, semua batasan rinci untuk setiap representasi berada di luar cakupan buku ini. Gambar 5-26 menggambarkan asosiasi di antara model struktural.

Pertanyaan

1. Jelaskan kepada seorang pebisnis multiplisitas hubungan antara dua kelas.
2. Mengapa asumsi penting untuk model struktural?
3. Apa itu kelas asosiasi?
4. Bandingkan set istilah berikut: objek, kelas, metode, atribut, superclass, subclass, kelas konkret, kelas abstrak.
5. Berikan tiga contoh atribut turunan yang mungkin ada pada diagram kelas. Bagaimana mereka akan dilambangkan pada diagram kelas?

6. Apa saja jenis-jenis visibilitas? Bagaimana mereka akan dilambangkan pada diagram kelas?
7. Gambarkan hubungan yang dijelaskan oleh aturan bisnis berikut. Sertakan multiplisitas untuk setiap hubungan.

   Seorang pasien harus ditugaskan hanya untuk satu dokter, dan seorang dokter dapat memiliki satu atau banyak pasien.

   Seorang karyawan memiliki satu ekstensi telepon, dan ekstensi telepon unik diberikan kepada seorang karyawan.

   Sebuah bioskop menampilkan setidaknya satu film, dan sebuah film dapat diputar di hingga empat bioskop lain di sekitar kota.

   Sebuah film memiliki satu bintang, dua lawan main, atau lebih dari sepuluh orang yang dibintangi bersama.

   Seorang bintang harus ada di setidaknya satu film.

8. Bagaimana Anda menentukan arah pembacaan hubungan pada diagram kelas?
9. Untuk apa kelas asosiasi digunakan dalam diagram kelas? Berikan contoh kelas asosiasi yang dapat ditemukan dalam diagram kelas yang menangkap siswa dan mata kuliah yang telah mereka ambil.
10. Berikan dua contoh hubungan agregasi, generalisasi, dan asosiasi. Bagaimana setiap jenis asosiasi digambarkan pada diagram kelas?
11. Identifikasi operasi berikut sebagai konstruktor, kueri, atau pembaruan. Operasi mana yang tidak perlu ditampilkan dalam persegi panjang kelas?

    Hitung kenaikan gaji karyawan (kenaikan persen)

    Hitung hari sakit ()

    Pertambahan jumlah hari libur karyawan ()

    Cari nama karyawan ()

    Tempatkan permintaan liburan (hari libur)

    Cari alamat karyawan ()

    Masukkan karyawan ()

    Ubah alamat karyawan ()

    Masukkan pasangan ()

12. Bagaimana model struktural yang berbeda terkait, dan bagaimana hal ini mempengaruhi verifikasi dan validasi model?

Latihan

A. Buat kartu CRC untuk setiap kelas berikut: Film (judul, produser, durasi, sutradara, genre) Tiket (harga, dewasa atau anak, waktu tayang, film) Pelindung (nama, dewasa atau anak, usia)
B. Buat diagram kelas berdasarkan kartu CRC yang Anda buat untuk latihan A.
C. Buat kartu CRC untuk setiap kelas berikut. Pertimbangkan bahwa entitas mewakili sistem untuk sistem penagihan pasien. Sertakan hanya atribut yang sesuai untuk

konteks ini. Pasien (usia, nama, hobi, golongan darah, pekerjaan, operator asuransi, alamat, telepon) Operator asuransi (nama, jumlah pasien dalam paket, alamat, nama kontak, telepon) Dokter (spesialis, nomor identifikasi penyedia, handicap golf, usia , telepon, nama)

D. Buat diagram kelas berdasarkan kartu CRC yang Anda buat untuk latihan C.
E. Gambarlah diagram kelas untuk masing-masing situasi berikut:
    1. Setiap kali pasien baru dilihat untuk pertama kalinya, mereka mengisi formulir informasi pasien yang menanyakan nama, alamat, nomor telepon, dan operator asuransi mereka, yang disimpan dalam informasi pasien mengajukan. Pasien dapat mendaftar hanya dengan satu operator, tetapi mereka harus mendaftar untuk dilihat oleh dokter. Setiap kali pasien mengunjungi dokter, klaim asuransi dikirim ke operator untuk pembayaran. Klaim harus berisi informasi tentang kunjungan, seperti tanggal, tujuan, dan biaya. Adalah mungkin bagi seorang pasien untuk mengajukan dua klaim pada hari yang sama.
    2. Negara bagian Georgia tertarik untuk merancang sebuah sistem yang akan melacak para penelitinya. Informasi yang diminati meliputi nama peneliti, gelar, jabatan, nama universitas peneliti, lokasi universitas, pendaftaran universitas, dan minat penelitian peneliti. Peneliti berasosiasi dengan satu institusi, dan setiap peneliti memiliki beberapa kepentingan penelitian.
    3. Sebuah department store memiliki daftar pernikahan. Registri ini menyimpan informasi tentang pelanggan (biasanya pengantin wanita), produk yang dibawa toko, dan produk yang didaftarkan oleh setiap pelanggan. Pelanggan biasanya mendaftar untuk sejumlah besar produk, dan banyak pelanggan mendaftar untuk produk yang sama.
    4. Dealer Jim Smith menjual Ford, Honda, dan Toyota. Untuk menghubungi produsen ini dengan mudah, dealer menyimpan informasi tentang masing-masing produsen. Dealer menyimpan informasi tentang model mobil dari masing-masing pabrikan, termasuk harga dealer, nama model, dan seri (misalnya, Honda, Civic, LX). Selain itu, dealer juga menyimpan semua informasi penjualan, termasuk nama pembeli, alamat dan nomor telepon, mobil yang dibeli, dan jumlah yang dibayarkan.
F. Buat diagram objek berdasarkan diagram kelas yang Anda gambar untuk latihan F.
G. Periksa diagram kelas yang Anda buat untuk latihan. Bagaimana model akan berubah (jika ada) berdasarkan asumsi baru ini?
    1. Dua pasien memiliki nama depan dan belakang yang sama.
    2. Peneliti dapat berasosiasi dengan lebih dari satu institusi.
    3. Toko ingin melacak item pembelian.
    4. Banyak pembeli telah membeli beberapa mobil dari Jim dari waktu ke waktu karena dia adalah dealer yang baik.
H. Kunjungi situs web yang memungkinkan pelanggan memesan produk melalui Web (mis., Amazon.com). Buat model struktural (kartu CRC dan diagram kelas) yang harus dibutuhkan situs untuk mendukung proses bisnisnya. Sertakan kelas untuk menunjukkan informasi yang mereka butuhkan. Pastikan untuk menyertakan atribut dan operasi untuk mewakili jenis informasi yang mereka gunakan dan buat. Terakhir, gambarkan hubungan, buat asumsi tentang bagaimana kelas-kelas tersebut saling berhubungan.
I. Dengan menggunakan proses tujuh langkah yang dijelaskan dalam bab ini, buat model struktural (kartu CRC dan diagram kelas) untuk latihan C di Bab 4.
J. Lakukan langkah-langkah verifikasi dan validasi untuk model struktural yang dibuat untuk latihan J.
K. Dengan menggunakan proses tujuh langkah yang dijelaskan dalam bab ini, buat model struktural untuk latihan E di Bab 4.

L. Lakukan langkah-langkah verifikasi dan validasi untuk model struktural yang dibuat untuk latihan L.
M. Dengan menggunakan proses tujuh langkah yang dijelaskan dalam bab ini, buat model struktural untuk latihan G di Bab 4.
N. Lakukan langkah verifikasi dan validasi untuk model struktural yang dibuat untuk latihan N. O. Dengan menggunakan proses tujuh langkah yang dijelaskan dalam bab ini, buat model struktural untuk latihan I di Bab 4.
O. Lakukan langkah-langkah verifikasi dan validasi untuk model struktural yang dibuat untuk latihan P.
P. Dengan menggunakan proses tujuh langkah yang dijelaskan dalam bab ini, buat model struktural untuk latihan L di Bab 4.
Q. Lakukan langkah-langkah verifikasi dan validasi untuk model struktural yang dibuat untuk latihan R.
R. Dengan menggunakan proses tujuh langkah yang dijelaskan dalam bab ini, buat model struktural untuk latihan O di Bab 4.
S. Lakukan langkah-langkah verifikasi dan validasi untuk model struktural yang dibuat untuk latihan T.
T. Dengan menggunakan proses tujuh langkah yang dijelaskan dalam bab ini, buat model struktural untuk latihan R di Bab 4.
U. Lakukan langkah-langkah verifikasi dan validasi untuk model struktural yang dibuat untuk latihan V.
V. Dengan menggunakan proses tujuh langkah yang dijelaskan dalam bab ini, buat model struktural untuk latihan U di Bab 4.
W. Lakukan langkah-langkah verifikasi dan validasi untuk model struktural yang dibuat untuk latihan X.

Minicase

1. West Star Marinas adalah rantai dua belas marina yang menawarkan layanan tepi laut untuk pelaut; pelayanan dan perbaikan kapal, motor, dan peralatan kelautan; dan penjualan perahu, motor, dan aksesoris kelautan lainnya. Tim proyek pengembangan sistem di West Star Marinas telah bekerja keras pada sebuah proyek yang pada akhirnya akan menghubungkan semua fasilitas marina menjadi satu sistem jaringan terpadu.

   Tim proyek telah mengembangkan diagram Use-Case dari sistem saat ini. Model ini telah diperiksa dengan cermat. Minggu lalu, tim mengundang sejumlah pengguna sistem untuk memainkan berbagai Use-Case, dan Use-Case disempurnakan untuk kepuasan pengguna. Saat ini, manajer proyek merasa yakin bahwa sistem saat in telah terwakili secara memadai dalam diagram Use-Case.

   Direktur operasi West Star adalah sponsor proyek ini. Dia ikut serta dalam permainan peran Use-Case dan sangat senang dengan pekerjaan menyeluruh yang telah dilakukan tim dalam mengembangkan model. Dia menjelaskan kepada Anda, manajer proyek, bahwa dia sangat ingin melihat tim Anda mulai mengerjakan Use-Case untuk sistem yang akan datang. Dia sedikit skeptis bahwa tim Anda perlu meluangkan waktu untuk memodelkan sistem saat ini, tetapi dengan enggan mengakui bahwa tim tampaknya benar-benar memahami bisnis setelah melalui pekerjaan itu.

   Metodologi yang Anda ikuti, bagaimanapun, menetapkan bahwa tim sekarang harus mengalihkan perhatiannya untuk mengembangkan model struktural untuk sistem apa adanya. Ketika Anda menyatakan hal ini kepada sponsor proyek, dia tampak bingung dan sedikit kesal. “Kamu akan menghabiskan lebih banyak waktu untuk melihat sistem

saat ini? Saya pikir Anda sudah selesai dengan itu! Mengapa ini perlu? Saya ingin melihat beberapa kemajuan tentang cara kerja sesuatu di masa depan!"

Apa tanggapan Anda terhadap direktur operasi? Mengapa kita melakukan pemodelan struktural? Apakah ada manfaat untuk mengembangkan model struktural dari sistem saat ini sama sekali? Bagaimana use case dan use case diagram membantu kita mengembangkan model struktural?

2. Holiday Travel Vehicles menjual kendaraan rekreasi dan trailer perjalanan baru. Ketika kendaraan baru tiba di Holiday Travel Vehicles, rekor kendaraan baru dibuat. Termasuk dalam catatan kendaraan baru adalah nomor seri kendaraan, nama, model, tahun, pabrikan, dan biaya dasar.

   Ketika pelanggan tiba di Holiday Travel Vehicles, dia bekerja dengan seorang penjual untuk menegosiasikan pembelian kendaraan. Ketika pembelian telah disepakati, faktur penjualan diisi oleh penjual. Faktur meringkas pembelian, termasuk informasi lengkap pelanggan, informasi kendaraan tukar tambah (jika ada), tunjangan tukar tambah, dan informasi kendaraan yang dibeli. Jika pelanggan meminta opsi yang dipasang dealer, opsi tersebut juga tercantum pada faktur. Faktur juga merangkum harga akhir yang dinegosiasikan, ditambah pajak dan biaya lisensi yang berlaku. Transaksi diakhiri dengan tanda tangan pelanggan pada faktur penjualan.

   1. Identifikasi kelas yang dijelaskan dalam skenario sebelumnya (Anda harus menemukan enam). Buat kartu CRC untuk setiap kelas.

      Pelanggan diberikan ID pelanggan saat mereka melakukan pembelian pertama dari Kendaraan Perjalanan Liburan. Nama, alamat, dan nomor telepon dicatat untuk pelanggan. Kendaraan tukar tambah dijelaskan dengan nomor seri, merek, model, dan tahun. Opsi yang dipasang dealer dijelaskan oleh kode opsi, deskripsi, dan harga.

   2. Kembangkan daftar atribut untuk setiap kelas. Tempatkan atribut ke kartu CRC.

      Setiap faktur hanya mencantumkan satu pelanggan. Seseorang tidak menjadi pelanggan sampai dia membeli kendaraan. Seiring waktu, pelanggan dapat membeli sejumlah kendaraan dari Kendaraan Perjalanan Liburan.

      Setiap faktur harus diisi hanya oleh satu tenaga penjual. Seorang wiraniaga baru mungkin tidak menjual kendaraan apa pun, tetapi wiraniaga berpengalaman mungkin telah menjual banyak kendaraan.

      Setiap faktur hanya mencantumkan satu kendaraan baru. Jika kendaraan baru dalam inventaris belum terjual, tidak akan ada faktur untuk itu. Setelah kendaraan terjual, hanya akan ada satu faktur untuk itu.

      Pelanggan dapat memutuskan untuk tidak memiliki opsi yang ditambahkan ke kendaraan atau dapat memilih untuk menambahkan banyak opsi. Sebuah pilihan mungkin terdaftar pada tidak ada faktur, atau mungkin terdaftar di banyak faktur.

      Pelanggan dapat memperdagangkan tidak lebih dari satu kendaraan untuk pembelian kendaraan baru. Kendaraan tukar tambah dapat dijual kepada pelanggan lain yang kemudian menukarkannya dengan kendaraan Perjalanan Liburan lainnya.

3. Berdasarkan aturan bisnis sebelumnya yang berlaku di Holiday Travel Vehicles dan kartu CRC, gambarlah diagram kelas dan dokumentasikan hubungan dengan multiplisitas yang sesuai. Ingatlah untuk memperbarui kartu CRC.

BAB 6

PERMODELAN PERILAKU

Model perilaku menggambarkan aspek dinamis internal dari sistem informasi yang mendukung proses bisnis dalam suatu organisasi. Selama analisis, model perilaku menggambarkan apa logika internal dari proses tanpa menentukan bagaimana proses akan diimplementasikan. Kemudian, dalam fase desain dan implementasi, desain rinci dari operasi yang terkandung dalam objek sepenuhnya ditentukan. Dalam bab ini, kami menjelaskan tiga diagram Unified Modeling Language (UML) yang digunakan dalam pemodelan perilaku (diagram urutan, diagram komunikasi, dan mesin status perilaku) dan matriks CRUDE (membuat, membaca, memperbarui, menghapus, mengeksekusi).

6.1 Tujuan

- Memahami aturan dan pedoman gaya untuk diagram urutan dan komunikasi serta mesin status perilaku.
- Memahami proses yang digunakan untuk membuat diagram urutan dan komunikasi, mesin status perilaku, dan matriks CRUDE.
- Mampu membuat diagram urutan dan komunikasi, mesin status perilaku, dan matriks CRUDE.
- Memahami hubungan antara model perilaku dan model struktural dan fungsional.

6.2 Pendahuluan

Dua bab sebelumnya membahas bagaimana analis membuat proses bisnis dan model fungsional dan model struktural. Analis sistem menggunakan proses bisnis dan model fungsional untuk menggambarkan pandangan perilaku fungsional atau eksternal dari sistem informasi. Dan, mereka menggunakan model struktural untuk menggambarkan pandangan struktural atau statis internal dari sistem informasi. Dalam bab ini, kita membahas bagaimana analis menggunakan model perilaku untuk mewakili perilaku internal atau pandangan dinamis dari sistem informasi.

Dengan mendukung ketiga pandangan (fungsional, struktural, dan perilaku), analisis dan desain sistem berorientasi objek mendukung pendekatan arsitektur-sentris untuk mengembangkan sistem informasi. Selanjutnya, pandangan perilaku didorong oleh Use-Case asli yang ditemukan selama proses bisnis dan pemodelan fungsional. Dengan demikian, pemodelan perilaku juga menggunakan kasus didorong. Akhirnya, seperti halnya proses bisnis dan pemodelan fungsional dan pemodelan struktural, Anda akan menemukan bahwa Anda tidak hanya perlu mengulangi seluruh model perilaku (dijelaskan dalam bab ini), tetapi Anda juga harus beralih di ketiga tampilan arsitektur (fungsional, struktural, dan perilaku) untuk menangkap dan mewakili persyaratan untuk sistem informasi bisnis.

Ada dua jenis model perilaku. Pertama, ada model perilaku yang digunakan untuk mewakili detail mendasar dari proses bisnis yang digambarkan oleh model Use-Case. Dalam UML, diagram interaksi (urutan dan komunikasi) digunakan untuk jenis model perilaku ini. Secara praktis, diagram interaksi memungkinkan analis untuk memodelkan distribusi perilaku sistem di atas aktor dan objek dalam sistem. Dengan cara ini, kita dapat dengan mudah melihat bagaimana aktor dan objek berkolaborasi untuk menyediakan fungsionalitas yang ditentukan dalam use case. Kedua, model perilaku digunakan untuk mewakili perubahan yang terjadi pada data yang mendasarinya. UML menggunakan mesin status perilaku untuk ini.

Selama analisis, analis menggunakan model perilaku untuk menangkap pemahaman dasar tentang aspek dinamis dari proses bisnis yang mendasarinya. Secara tradisional, model perilaku telah digunakan terutama selama desain, di mana analis menyempurnakan model perilaku untuk memasukkan rincian implementasi (lihat Bab 8). Untuk saat ini, fokus kami adalah pada apa pandangan dinamis dari sistem yang berkembang dan bukan pada bagaimana aspek dinamis dari sistem akan diimplementasikan.

Dalam bab ini, kami berkonsentrasi pada pembuatan model perilaku dari proses bisnis yang mendasarinya. Dengan menggunakan diagram interaksi (diagram urutan dan komunikasi) dan mesin status perilaku, dimungkinkan untuk memberikan pandangan lengkap tentang aspek dinamis dari sistem informasi bisnis yang berkembang. Kami pertama-tama menjelaskan model perilaku dan komponennya. Kami kemudian menjelaskan masing-masing diagram, bagaimana mereka dibuat, dan bagaimana mereka terkait dengan model fungsional dan struktural yang dijelaskan dalam Bab 4 dan 5. Akhirnya, kami menjelaskan analisis CRUDE dan proses untuk memverifikasi dan memvalidasi model perilaku.

6.3 MODEL PERILAKU

Ketika seorang analis mencoba untuk memahami domain aplikasi yang mendasari suatu masalah, dia harus mempertimbangkan aspek struktural dan perilaku dari masalah tersebut. Tidak seperti pendekatan lain untuk pengembangan sistem informasi, pendekatan berorientasi objek mencoba untuk melihat domain aplikasi yang mendasarinya secara holistik. Dengan melihat domain masalah sebagai satu set Use-Case yang didukung oleh satu set objek yang berkolaborasi, pendekatan berorientasi objek memungkinkan seorang analis untuk meminimalkan kesenjangan semantik antara set objek dunia nyata dan model berorientasi objek yang berkembang dari domain masalah. Namun, seperti yang kami tunjukkan di bab sebelumnya, dunia nyata cenderung berantakan; karena perangkat lunak harus logis untuk bekerja, pemodelan sempurna dari domain aplikasi hampir tidak mungkin.

Salah satu tujuan utama dari model perilaku adalah untuk menunjukkan bagaimana objek yang mendasari dalam domain masalah akan bekerja sama untuk membentuk kolaborasi untuk mendukung setiap Use-Case. Sementara model struktural mewakili objek dan hubungan di antara mereka, model perilaku menggambarkan pandangan internal dari proses bisnis yang dijelaskan oleh Use-Case. Proses tersebut dapat ditunjukkan dengan interaksi yang terjadi antara objek-objek yang berkolaborasi untuk mendukung sebuah use case melalui penggunaan diagram interaksi (sequence and communication). Hal ini juga memungkinkan untuk menunjukkan efek yang dimiliki set Use-Case yang membentuk sistem pada objek dalam sistem melalui penggunaan mesin status perilaku.

Membuat model perilaku adalah proses berulang yang berulang tidak hanya pada model perilaku individu [misalnya, diagram interaksi (urutan dan komunikasi) dan mesin status perilaku] tetapi juga pada model fungsional (lihat Bab 4) dan struktural (lihat Bab 5). Ketika model perilaku dibuat, bukan hal yang aneh untuk membuat perubahan pada model fungsional dan struktural. Dalam bab ini, kami menjelaskan diagram interaksi, mesin status perilaku, dan analisis CRUDE dan kapan harus menggunakannya.

6.4 DIAGRAM INTERAKSI

Salah satu perbedaan utama antara diagram kelas dan diagram interaksi, selain perbedaan yang jelas yang menggambarkan struktur dan perilaku lainnya, adalah bahwa fokus pemodelan pada diagram kelas adalah pada tingkat kelas, sedangkan diagram interaksi fokus pada tingkat objek. Pada bagian ini, kami meninjau objek, operasi, dan pesan dan kami membahas dua diagram berbeda (urutan dan komunikasi) yang dapat digunakan untuk memodelkan interaksi yang terjadi antara objek dalam sistem informasi.

Objek, Operasi, dan Pesan

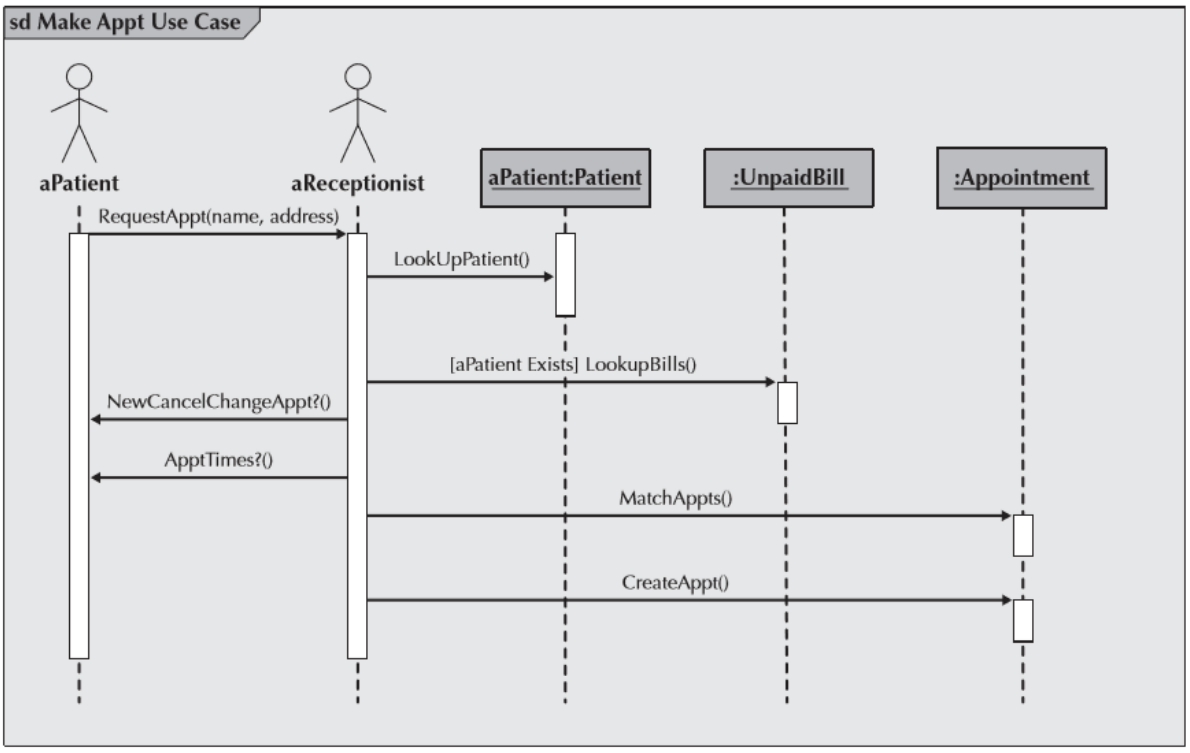
Objek adalah instantiasi kelas, yaitu, orang, tempat, atau benda aktual yang ingin kita tangkap informasinya. Jika kami sedang membangun sistem janji temu untuk kantor dokter, kelas mungkin termasuk dokter, pasien, dan janji temu. Pasien tertentu, seperti Jim Maloney, Mary Wilson, dan Theresa Marks, dianggap sebagai objek—yaitu, contoh kelas pasien.

Setiap objek memiliki atribut yang menggambarkan informasi tentang objek tersebut, seperti nama pasien, tanggal lahir, alamat, dan nomor telepon. Setiap objek juga memiliki perilaku. Pada titik ini dalam pengembangan sistem yang berkembang, perilaku dijelaskan oleh operasi. Operasi tidak lebih dari tindakan yang dapat dilakukan oleh objek. Misalnya, objek janji temu mungkin dapat menjadwalkan janji temu baru, menghapus janji temu, dan menemukan janji temu berikutnya yang tersedia. Kemudian selama pengembangan sistem yang berkembang, perilaku akan diimplementasikan sebagai metode.

Setiap objek juga dapat mengirim dan menerima pesan. Pesan adalah informasi yang dikirim ke objek untuk memberitahu objek untuk mengeksekusi salah satu perilakunya. Pada dasarnya, pesan adalah panggilan fungsi atau prosedur dari satu objek ke objek lain. Misalnya, jika pasien baru masuk ke ruang praktik dokter, sistem akan mengirimkan pesan penyisipan ke aplikasi. Objek pasien menerima instruksi (pesan) dan melakukan apa yang perlu dilakukan untuk memasukkan pasien baru ke dalam sistem (perilaku).

Diagram Urutan

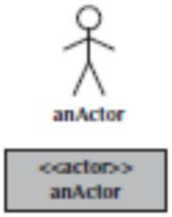
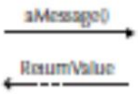
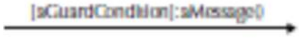
Diagram urutan adalah salah satu dari dua jenis diagram interaksi. Mereka mengilustrasikan objek yang berpartisipasi dalam use case dan pesan yang lewat di antara mereka dari waktu ke waktu untuk satu use case. Diagram urutan adalah model dinamis yang menunjukkan urutan eksplisit pesan yang dikirimkan antara objek dalam interaksi yang ditentukan. Karena diagram urutan menekankan pengurutan berdasarkan waktu dari aktivitas yang terjadi di antara sekumpulan objek, diagram ini sangat membantu untuk memahami spesifikasi waktu nyata dan Use-Case yang kompleks.



Gambar 6-1 Contoh Diagram Urutan

Diagram urutan dapat menjadi diagram urutan generik yang menunjukkan semua skenario yang mungkin untuk use case, tetapi biasanya setiap analis mengembangkan satu set diagram urutan instance, yang masing-masing menggambarkan satu skenario dalam use case. Jika Anda tertarik untuk memahami aliran kontrol skenario berdasarkan waktu, Anda harus menggunakan diagram urutan untuk menggambarkan informasi ini. Diagram digunakan selama fase analisis dan desain. Namun, diagram desain sangat spesifik untuk implementasi, sering kali menyertakan objek database atau komponen antarmuka pengguna tertentu sebagai objek.

Elemen Diagram Urutan. Gambar 6-1 menunjukkan diagram urutan instans yang menggambarkan objek dan pesan untuk Use-Case Make Old Patient Appt, yang menjelaskan proses di mana pasien yang sudah ada membuat janji temu baru atau membatalkan atau menjadwal ulang janji temu untuk sistem janji temu kantor dokter. Dalam contoh khusus ini, proses Make Old Patient Appt digambarkan.

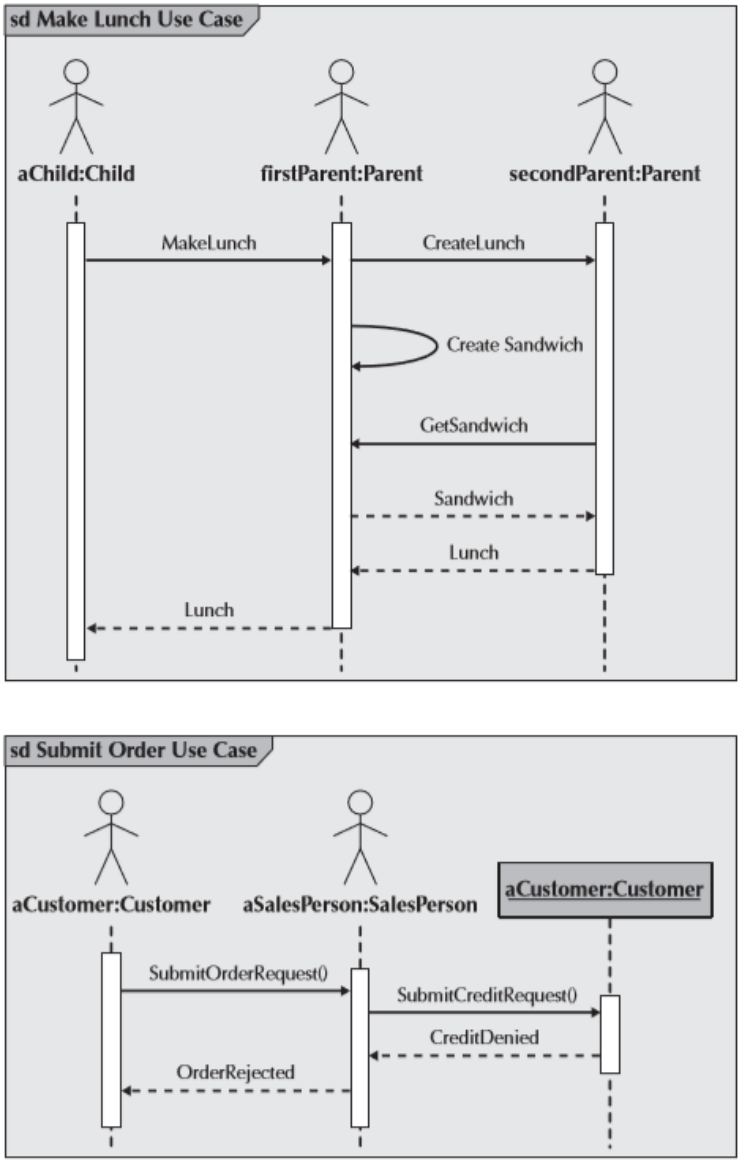
Istilah dan Definisi	Simbol
Aktor: ■ Adalah orang atau sistem yang mendapatkan peran di luar sistem. ■ Berpartisipasi dalam kolaborasi dengan mengirim dan/atau menerima pesan. ■ Ditempatkan di bagian atas diagram. ■ Digambarkan sebagai figur tongkat (default) atau, jika aktor nonmanusia terlibat, digambarkan persegi panjang dengan <<actor>> di dalamnya (alternatif).	
Objek: ■ Berpartisipasi dalam kolaborasi dengan mengirim dan/atau menerima pesan ■ Ditempatkan di bagian atas diagram.	
Lifeline: ■ Menunjukkan kehidupan suatu objek dalam urutan. ■ Berisi X pada titik di mana kelas tidak lagi berinteraksi.	
Eksekusi Peristiwa: ■ Adalah persegi panjang sempit yang ditempatkan di atas lifeline.. ■ Petunjuk ketika suatu objek mengirim atau menerima pesan.	
Pesan: ■ Menyampaikan informasi dari satu objek ke objek lainnya. ■ Panggilan operasi diberi label dengan pesan yang dikirim dan panah padat, sedangkan pengembalian diberi label dengan nilai yang dikembalikan dan ditampilkan dengan panah putus-putus.	
Kondisi Penjagaan: ■ Merupakan tes yang harus dipenuhi untuk pesan yang akan dikirim.	
Untuk penghancuran objek: ■ Sebuah X ditempatkan di akhir lifeline suatu objek untuk menunjukkan bahwa itu akan keluar dari keberadaan.	
Bingkai: ■ Menunjukkan konteks diagram urutan.	

Gambar 6-2 Sintaks Diagram Urutan

Aktor dan objek yang berpartisipasi dalam urutan ditempatkan di bagian atas diagram menggunakan simbol aktor dari diagram use-case dan simbol objek dari diagram objek (lihat Gambar 6-2). Perhatikan bahwa aktor dan objek pada Peraga 6-1 adalah aPatient, aReceptionist, aPatient, UnpaidBill, dan Appointment. Untuk setiap objek, nama kelas di mana mereka menjadi instance diberikan setelah nama objek (mis. aPatient berarti bahwa aPatient adalah turunan dari kelas Pasien).

Garis putus-putus berjalan vertikal di bawah setiap aktor dan objek untuk menunjukkan garis hidup aktor dan objek dari waktu ke waktu (lihat Gambar 6-1).Kadang-kadang sebuah objek membuat objek sementara; dalam hal ini, sebuah X ditempatkan di akhir garis hidup pada titik di mana objek dihancurkan (tidak ditampilkan). Misalnya, pikirkan tentang objek keranjang belanja untuk aplikasi perdagangan Web. Keranjang belanja digunakan untuk sementara menangkap item baris untuk pesanan, tetapi setelah pesanan dikonfirmasi, keranjang belanja tidak lagi diperlukan. Dalam hal ini, sebuah X akan

ditempatkan pada titik di mana objek keranjang belanja dihancurkan. Ketika objek terus ada dalam sistem setelah digunakan dalam diagram urutan, maka garis hidup berlanjut ke bagian bawah diagram (ini adalah kasus dengan semua objek pada Gambar 6-1).



Gambar 6-3 Contoh Tambahan Diagram Urutan Khusus-Instance

Kotak persegi panjang tipis, yang disebut kejadian eksekusi, dilapiskan ke garis hidup untuk menunjukkan kapan kelas mengirim dan menerima pesan (lihat Gambar 6-2). Pesan adalah komunikasi antara objek yang menyampaikan informasi dengan harapan bahwa aktivitas akan terjadi. Banyak jenis pesan yang berbeda dapat digambarkan pada diagram urutan. Namun, dalam Use-Case diagram urutan untuk memodelkan Use-Case, dua jenis pesan biasanya digunakan: panggilan operasi dan pengembalian. Pesan panggilan operasi yang diteruskan di antara objek ditampilkan menggunakan garis padat yang menghubungkan dua objek dengan panah di garis yang menunjukkan ke arah mana pesan tersebut diteruskan. Nilai argumen untuk pesan ditempatkan dalam tanda kurung di sebelah nama pesan. Urutan pesan dimulai dari atas ke bawah halaman, sehingga pesan yang terletak lebih tinggi pada diagram mewakili pesan yang muncul lebih awal dalam urutan, versus pesan yang lebih rendah yang muncul kemudian. Pesan kembali digambarkan sebagai garis putus-putus dengan panah di ujung garis yang menggambarkan arah pengembalian. Informasi yang dikembalikan digunakan untuk memberi label panah. Namun, karena menambahkan pesan kembali

cenderung mengacaukan diagram, kecuali jika pesan kembali menambahkan banyak informasi ke diagram, pesan tersebut dapat dihilangkan. Misalnya, pada Gambar 6-1, tidak ada pesan balasan yang digambarkan. Pada Gambar 6-1, LookUpPatient() adalah pesan yang dikirim dari aktor aReceptionist ke objek aPatient untuk menentukan apakah aktor aPatient adalah pasien saat ini.

Kadang-kadang pesan dikirim hanya jika suatu kondisi terpenuhi. Dalam kasus tersebut, kondisi ditempatkan di antara sekumpulan tanda kurung, [ ]—misalnya, [aPatient Exists] LookupBills(). Kondisi ditempatkan di depan nama pesan. Namun, saat menggunakan diagram urutan untuk memodelkan skenario tertentu, kondisi biasanya tidak ditampilkan pada diagram urutan tunggal. Sebaliknya, kondisi tersirat hanya melalui keberadaan diagram urutan yang berbeda.

Sebuah objek dapat mengirim pesan ke dirinya sendiri, misalnya Create Sandwich pada Gambar 6-3. Ini dikenal sebagai pendelegasian diri. Terkadang, sebuah objek menciptakan objek lain. Ini ditunjukkan oleh pesan yang dikirim langsung ke suatu objek, bukan garis hidupnya.

Gambar 6-3 menggambarkan dua contoh tambahan diagram urutan spesifik-instance. Yang pertama terkait dengan use case Make Lunch yang dijelaskan dalam diagram aktivitas yang digambarkan pada Gambar 4-10. Yang kedua terkait dengan use case Place Order yang terkait dengan diagram aktivitas pada Gambar 4-9. Dalam kedua contoh, diagram hanya mewakili satu skenario. Perhatikan dalam diagram urutan Make Lunch ada pesan yang dikirim dari aktor ke dirinya sendiri [CreateSandwich()]. Bergantung pada kompleksitas skenario yang dimodelkan, pesan khusus ini bisa saja dihilangkan. Jelas, proses membuat makan siang dan memesan bisa sedikit lebih rumit. Namun, dari sudut pandang pembelajaran, Anda harus dapat melihat bagaimana diagram urutan dan diagram aktivitas berhubungan satu sama lain.

Pedoman untuk Membuat Diagram Urutan, Ambler menyediakan seperangkat pedoman saat menggambar diagram urutan. Di bagian ini, kami mengulas enam di antaranya.

- Usahakan agar pesan tidak hanya dalam urutan atas-ke-bawah tetapi juga, jika memungkinkan, dalam urutan kiri-ke-kanan. Mengingat bahwa budaya Barat cenderung membaca dari kiri ke kanan dan dari atas ke bawah, diagram urutan jauh lebih mudah untuk ditafsirkan jika pesan-pesan diurutkan sebanyak mungkin dengan cara yang sama. Untuk mencapai ini, urutkan aktor dan objek di sepanjang bagian atas diagram agar mereka berpartisipasi dalam skenario use case.
- Jika aktor dan objek secara konseptual mewakili ide yang sama, satu di dalam perangkat lunak dan yang lain di luar, beri label dengan nama yang sama. Faktanya, ini menyiratkan bahwa mereka ada di diagram use-case (sebagai aktor) dan di diagram kelas (sebagai kelas). Pada pandangan pertama, ini mungkin tampak menyebabkan kebingungan. Namun, jika mereka memang mewakili ide yang sama, maka mereka harus memiliki nama yang sama. Misalnya, aktor pelanggan berinteraksi dengan sistem dan sistem menyimpan informasi tentang pelanggan. Dalam hal ini, mereka memang mewakili ide konseptual yang sama.
- Pemrakarsa skenario—aktor atau objek—harus digambar sebagai item paling kiri dalam diagram. Pedoman ini pada dasarnya merupakan spesialisasi dari pedoman pertama. Dalam hal ini berkaitan secara khusus dengan aktor atau objek yang memicu skenario.
- Jika ada beberapa objek dengan tipe yang sama, pastikan untuk menyertakan nama objek selain kelas objek. Misalnya, dalam membuat contoh makan siang (lihat Gambar 6-3) ada dua objek bertipe Parent. Karena itu, mereka harus diberi nama. Jika tidak, Anda cukup menggunakan nama kelas. Ini akan menyederhanakan diagram. Dalam

hal ini, objek Anak tidak harus diberi nama. Kita bisa saja menyiratkan menempatkan titik dua di depan nama kelas sebagai gantinya.

- Tampilkan nilai pengembalian hanya jika tidak jelas. Menampilkan semua pengembalian cenderung membuat diagram urutan lebih kompleks dan berpotensi sulit untuk dipahami. Dalam banyak kasus, lebih sedikit lebih banyak. Hanya tampilkan pengembalian yang benar-benar menambah informasi bagi pembaca diagram.
- Ratakan nama pesan dan kembalikan nilai masing-masing di dekat panah pesan dan panah kembali. Ini membuatnya lebih mudah untuk menafsirkan pesan dan nilai pengembaliannya.

Membuat Diagram Urutan Pada bagian ini, kami menjelaskan proses enam langkah yang digunakan untuk membuat diagram urutan. Langkah pertama dalam proses ini adalah menentukan konteks diagram urutan. Konteks diagram dapat berupa sistem, use case, atau skenario use case. Konteks diagram digambarkan sebagai bingkai berlabel di sekitar diagram (lihat Gambar 6-1, 6-2, dan 6-3). Paling umum, ini adalah satu skenario Use-Case. Gambar 6-1 menggambarkan diagram urutan spesifik-instance untuk skenario dari Use-Case Make Old Patient Appt yang diberikan pada Gambar 4-13 untuk membuat janji temu baru untuk pasien yang sudah ada. Untuk setiap skenario yang mungkin untuk Use-Case Make Old Patient Appt, diagram urutan spesifik-instance terpisah akan dibuat. Di permukaan, ini tampaknya merupakan pekerjaan yang berpotensi berlebihan dan tidak berguna. Namun, pada titik ini dalam representasi suatu sistem, kami masih mencoba untuk memahami masalahnya sepenuhnya. Proses pembuatan diagram urutan khusus-instance ini untuk setiap skenario alih-alih membuat diagram urutan generik tunggal untuk seluruh Use-Case akan memungkinkan pengembang untuk mencapai pemahaman yang lebih lengkap tentang masalah yang sedang ditangani. Setiap diagram urutan spesifik-instance cukup sederhana untuk ditafsirkan, sedangkan diagram urutan generik bisa sangat kompleks. Pengujian Use-Case tertentu dicapai dengan cara yang jauh lebih mudah dengan memvalidasi dan memverifikasi kelengkapan rangkaian diagram urutan khusus-instance daripada mencoba bekerja melalui diagram urutan generik tunggal yang kompleks.

Langkah kedua adalah mengidentifikasi aktor dan objek yang berpartisipasi dalam urutan yang dimodelkan — yaitu, aktor dan objek yang berinteraksi satu sama lain selama skenario use-case. Aktor diidentifikasi selama pembuatan model fungsional, sedangkan objek diidentifikasi selama pengembangan model struktural. Ini adalah kelas di mana objek diagram urutan untuk skenario ini akan didasarkan. Salah satu pendekatan yang sangat berguna untuk mengidentifikasi semua skenario yang terkait dengan use case adalah dengan memainkan kartu CRC (lihat Bab 5). Ini dapat membantu Anda mengidentifikasi operasi yang berpotensi hilang yang diperlukan untuk mendukung proses bisnis, yang diwakili oleh kasus penggunaan, secara lengkap. Juga, selama bermain peran, kemungkinan kelas baru, dan karenanya objek baru, akan terungkap. Jangan terlalu khawatir tentang mengidentifikasi semua objek dengan sempurna; ingat bahwa proses pemodelan perilaku itu berulang. Biasanya, diagram urutan direvisi beberapa kali selama proses pemodelan perilaku.

Langkah ketiga adalah mengatur garis hidup untuk setiap objek. Untuk melakukan ini, Anda perlu menggambar garis putus-putus vertikal di bawah setiap kelas untuk mewakili keberadaan kelas selama urutan. Sebuah X harus ditempatkan di bawah objek pada titik pada garis kehidupan di mana objek tersebut menghilang.

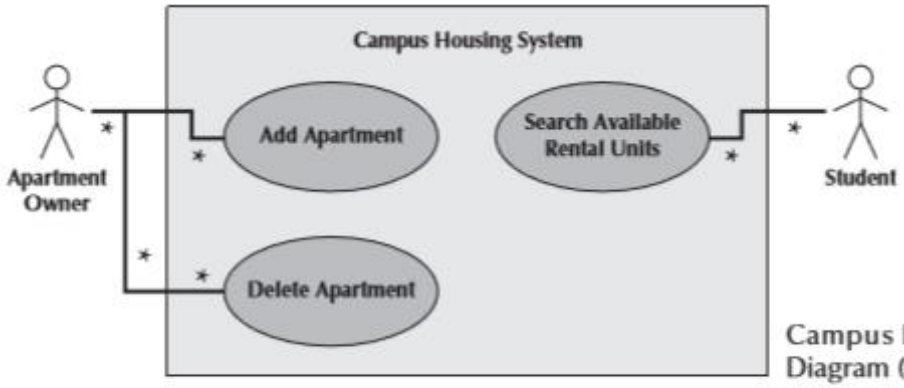
Langkah keempat adalah menambahkan pesan ke diagram. Ini dilakukan dengan menggambar panah untuk mewakili pesan yang dikirimkan dari objek ke objek, dengan panah menunjuk ke arah transmisi pesan. Panah harus ditempatkan secara berurutan dari pesan pertama (di atas) hingga yang terakhir (di bawah) untuk menunjukkan urutan waktu.

Parameter apa pun yang diteruskan bersama pesan harus ditempatkan dalam tanda kurung di sebelah nama pesan. Jika sebuah pesan diharapkan untuk dikembalikan sebagai respons terhadap sebuah pesan, maka pesan yang dikembalikan tidak secara eksplisit ditampilkan pada diagram.

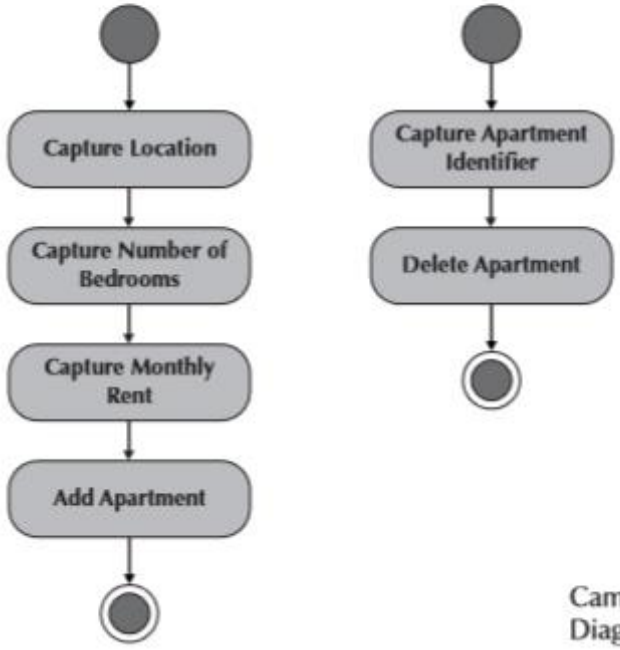
Langkah kelima adalah menempatkan kejadian eksekusi pada garis hidup setiap objek dengan menggambar kotak persegi panjang sempit di atas garis hidup untuk mewakili kapan kelas mengirim dan menerima pesan.

Langkah keenam dan terakhir adalah memvalidasi sequence diagram. Tujuan dari langkah ini adalah untuk menjamin bahwa diagram urutan benar-benar mewakili proses yang mendasarinya. Hal ini dilakukan dengan menjamin bahwa diagram menggambarkan semua langkah dalam proses.

Contoh Perumahan Kampus Dalam Bab 4 dan 5, kami membuat satu set model fungsional dan struktural untuk layanan perumahan kampus. Di bagian ini, kita akan menggunakan model tersebut untuk membuat diagram urutan untuk Use-Case Tambah Apartemen. Seperti yang dinyatakan di atas, hal pertama yang harus kita lakukan adalah mengatur konteksnya, yang dalam hal ini adalah use case Tambah Apartemen. Kedua, kita harus mengidentifikasi aktor dan objek yang akan berpartisipasi dalam eksekusi use case. Untuk melakukan ini, kita harus meninjau model fungsional dan struktural yang dibuat untuk masalah layanan perumahan kampus di Bab 4 dan 5. Gambar 6-4 mereplikasi representasi ini.



Campus Housing Use-Case Diagram (Figure 4-15)



Campus Add and Delete Apartment Activity Diagrams (Figure 4-16)

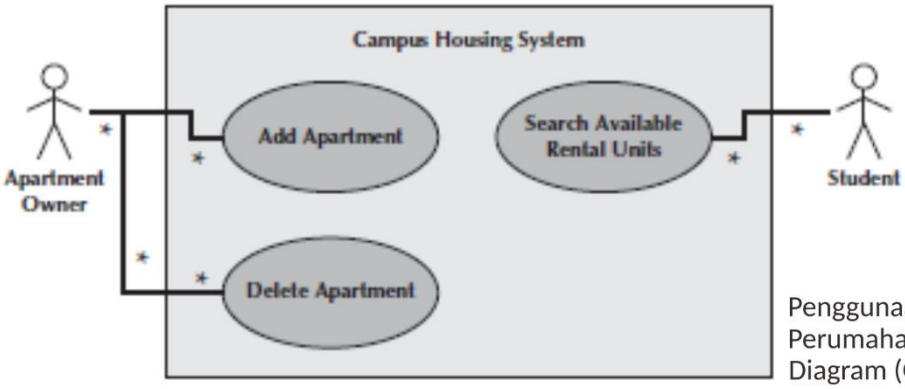
Nama Use Case : Add Apartment	ID : 1	Level Kesulitan : Tinggi
Primary Actor: Apartment Owner Use-Case Type: Detail, Essential		
Stakeholders and Interests: Pemilik Apartemen – ingin mengiklankan apartemen yang tersedia Layanan Perumahan Kampus – menyediakan layanan yang memungkinkan pemilik apartemen untuk menyewa apartemen yang tersedia		
Deskripsi Singkat: Use-Case ini menjelaskan bagaimana layanan perumahan kampus dapat mempertahankan daftar terbaru dari apartemen yang tersedia.		
Pemicu: Pemilik Apartemen ingin menambahkan apartemen yang tersedia. Type : External		
Relationships: Association: Sales Rep Include: Extend: Generalization:		
Alur Peristiwa Normal: 7. Menangkap lokasi apartemen. 8. Menangkap jumlah kamar tidur di apartemen. 9. Menangkap sewa bulanan apartemen. 10. Tambahkan apartemen ke daftar apartemen yang tersedia.		
Subflow:		
Aliran Alternatif/Luar Biasa:		

Layanan Perumahan Kampus Tambahkan Deskripsi Use-Case Apartemen (Gambar 4-17)

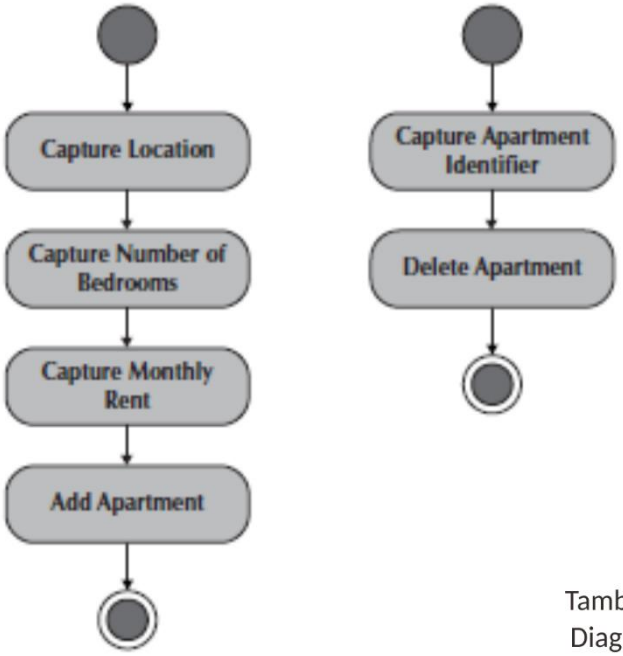
Nama Use Case : Delete Apartment	ID : 2	Level Kesulitan : Tinggi
Primary Actor: Apartment Owner Use-Case Type: Detail, Essential		
Stakeholders and Interests: Pemilik Apartemen – ingin menghapus apartemen Layanan Perumahan Kampus – menyediakan layanan yang memungkinkan pemilik apartemen untuk menyewa apartemen yang tersedia.		
Deskripsi Singkat: Use-Case ini menjelaskan bagaimana layanan perumahan kampus dapat mempertahankan daftar terbaru dari apartemen yang tersedia..		

Pemicu: Pemilik Apartemen ingin menghapus apartemen yang tersedia. Type : External
Relationships: Association: Sales Rep Include: Extend: Generalization:
Alur Peristiwa Normal: 1. Tangkap pengenal apartemen. 2. Hapus apartemen dari daftar apartemen yang tersedia
Subflow:
Aliran Alternatif/Luar Biasa:

Layanan Perumahan Kampus Hapus Deskripsi Use-Case Apartemen (Gambar 4-18)



Penggunaan Kasus Kampus Perumahan Diagram (Gambar 4-15)

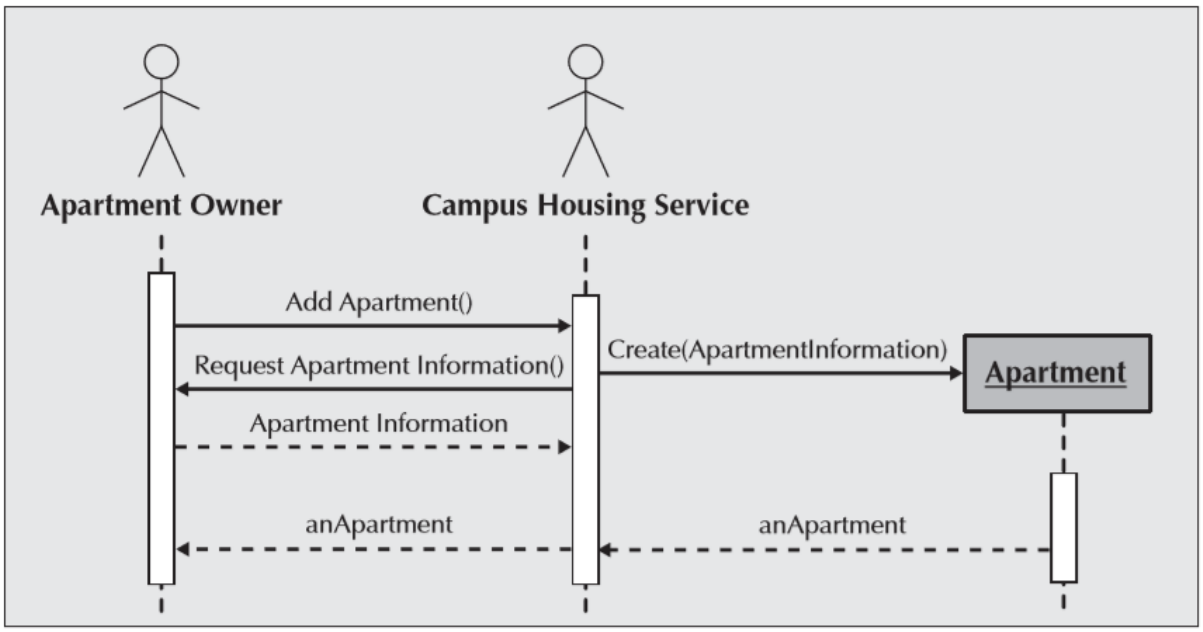


Tambah kampus dan Hapus Aktivitas Apartemen Diagram (Gambar 4-16)

[8] Kami menjelaskan validasi secara lebih rinci nanti di bab ini

Gambar 6-4 Model Fungsional dan Struktural Pelayanan Perumahan Kampus

Berdasarkan representasi fungsional dan struktural, terlihat bahwa aktor yang terlibat dalam use case adalah Pemilik Apartemen dan Dinas Perumahan Kampus itu sendiri. Dengan melihat melalui Normal Flow of Events dan diagram aktivitas, kita melihat bahwa satu-satunya objek yang tampaknya relevan adalah instance dari kelas Apartment yang ditambahkan. Mengingat bahwa tidak ada Alternate/Exceptional Flows atau keputusan apa pun yang dibuat dalam Normal Flow of Events, juga tidak ada keputusan dalam diagram aktivitas yang terkait dengan use case Add Apartment, hanya ada satu skenario yang akan digambarkan. Akibatnya, hanya ada satu diagram urutan khusus-instance yang akan dibuat. Gambar 6-5 menggambarkan diagram urutan untuk use case ini.



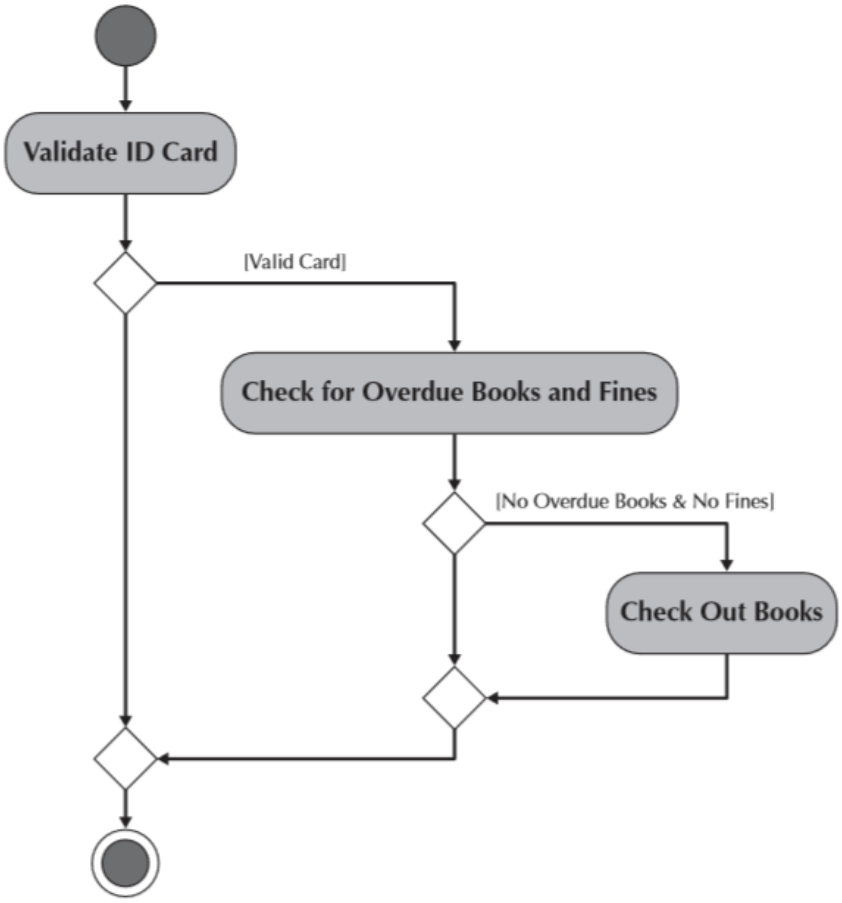
Gambar 6-5 Diagram Urutan untuk Use Case Add A part

Contoh Perpustakaan Pada bab-bab sebelumnya, kami telah mendemonstrasikan proses pembuatan diagram dan pemodelan menggunakan use case Pinjam Buku dari Sistem Manajemen Koleksi Buku Perpustakaan. Saat mempertimbangkan diagram skenario khusus instance, kita perlu menggambar satu diagram urutan per skenario. Dalam kasus use case Pinjam Buku di Bab 4, ada sembilan skenario yang berbeda. Oleh karena itu, untuk use case yang satu ini, akan ada sembilan diagram terpisah. Dalam contoh ini, kami menetapkan konteks diagram urutan ke hanya satu skenario spesifik dari Use-Case Pinjam Buku: Siswa yang memiliki ID yang valid dan tidak memiliki buku yang terlambat atau denda apa pun. Skenario lainnya termasuk Siswa tanpa ID yang valid, Siswa dengan ID yang valid tetapi yang berhutang denda atau memiliki buku yang terlambat, dan tiga skenario yang sama untuk dua jenis Peminjam lainnya: Fakultas/Staf dan Tamu. Dalam contoh ini, kami hanya menggambar diagram urutan satu untuk Siswa dengan skenario ID yang valid. Untuk memulainya, kita harus meninjau Flow of Events dari deskripsi use-case (lihat Gambar 6-6), diagram aktivitas (lihat Gambar 6-7), dan diagram use-case (lihat Gambar 6-8).

Alur Acara Normal: 1. Peminjam membawa buku ke Pustakawan di meja check out. 2. Peminjam memberikan Kartu Tanda Penduduk kepada Pustakawan. 3. Pustakawan memeriksa keabsahan KTP. Jika Peminjam adalah Peminjam Pelajar, Validasi KTP terhadap Database Panitera. Jika Peminjam adalah Peminjam Fakultas/Staf, Validasi KTP terhadap Database Kepegawaian. Jika Peminjam adalah Peminjam Tamu, Validasi KTP terhadap Database Tamu Perpustakaan. 4. Pustakawan memeriksa apakah Peminjam memiliki buku dan/atau denda yang jatuh tempo. 5. Peminjam memeriksa buku.
SubAliran:
Aliran Alternatif/Luar Biasa: 4a. ID Card tidak valid, permintaan buku ditolak.

5a. Peminjam memiliki buku yang lewat jatuh tempo, denda, atau keduanya, permintaan buku ditolak.

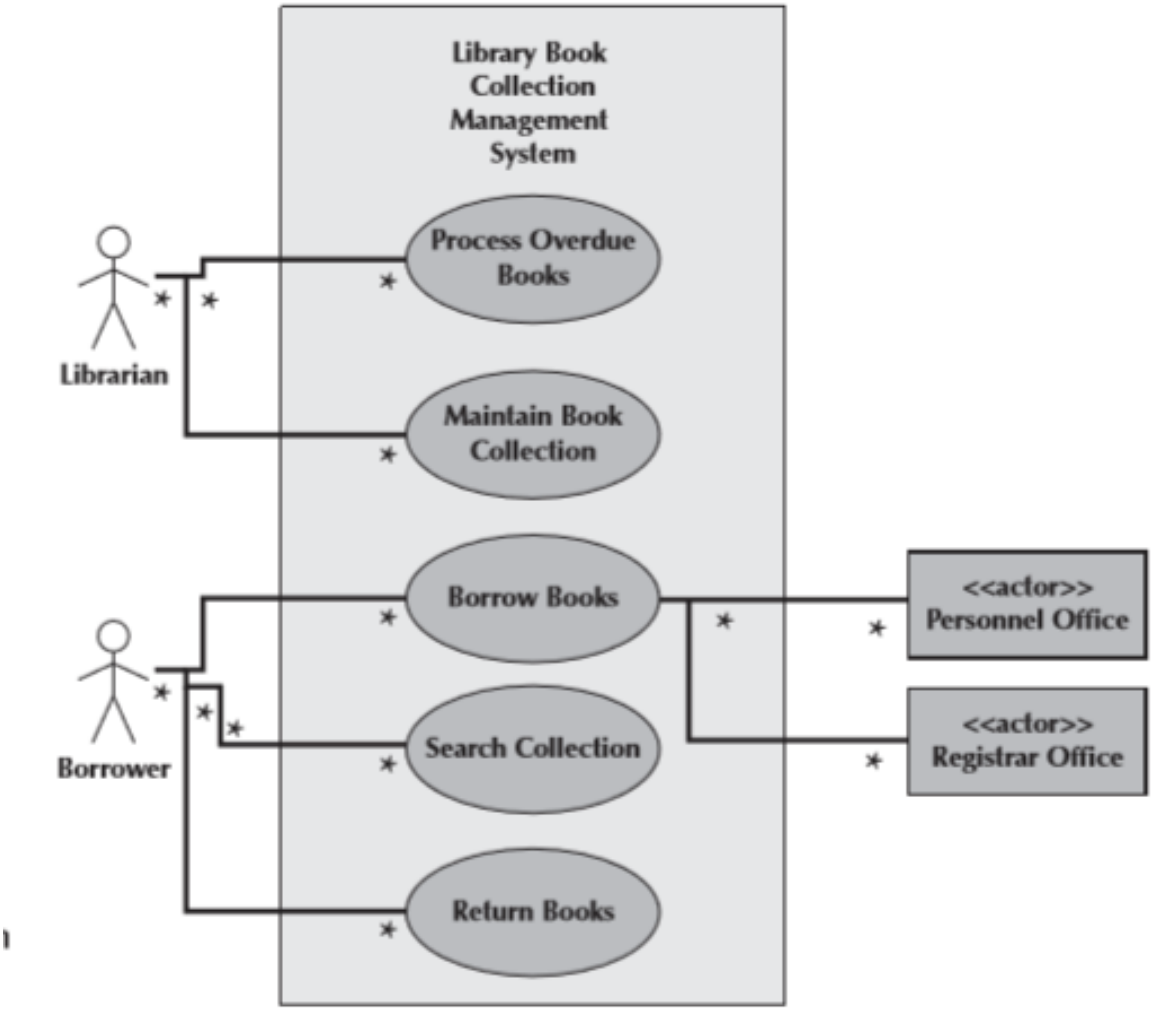
Gambar 6-6 Alur Peristiwa Bagian Use-Case Deskripsi Use Case Pinjam Buku



Gambar 6-7 Activity Diagram Use Case Pinjam Buku (Gambar 4-12)

Langkah selanjutnya adalah mengidentifikasi aktor dan objek yang terlibat dalam skenario. Dengan mempelajari alur peristiwa dan diagram use-case, kami mengidentifikasi siswa, pustakawan, dan database pendaftar sebagai aktor dan peminjam, koleksi buku, dan buku sebagai objek. Kami menempatkan aktor dan objek di bagian atas diagram berdasarkan urutan penampilan mereka dalam aliran peristiwa yang normal. Langkah selanjutnya melibatkan menggambar garis hidup di bawah aktor dan objek dalam skenario. Langkah keempat adalah menambahkan pesan yang sebenarnya ke diagram. Untuk melakukan ini, kami meninjau kembali langkah-langkah aktual yang diambil saat menjalankan skenario ini dengan meninjau alur kejadian (lihat Gambar 6-6) dan diagram aktivitas (lihat Gambar 6-7). Kita juga harus meninjau setiap hasil dari permainan peran kartu CRC (lihat Bab 5). Ini akan membantu kami untuk menggambarkan dengan benar di mana fungsi tersebut berada. Misalnya, pada Gambar 6-9, Pustakawan menjalankan prosedur CheckOutBooks() (Siswa mengirimkan pesan CheckOutBooks () untuk meminta Pustakawan menjalankan prosedur CheckOutBooks() saat siswa menyerahkan buku kepada pustakawan untuk diperiksa. Pustakawan sebagai gantinya meminta Kartu ID Siswa. Ketika siswa menyerahkan KTP kepada Pustakawan, Pustakawan meminta Database Panitera untuk menjalankan prosedur ValidID() ketika Pustakawan memberikan nomor ID siswa ke sistem database untuk meminta sistem database untuk memvalidasi nomor ID siswa. Hal ini berlanjut sampai KTP dan Buku dikembalikan ke siswa. Setelah kami memutuskan dari siapa pesan akan dikirim dan kepada

siapa mereka dikirim, kami dapat menempatkan pesan pada diagram. Langkah kelima kemudian adalah menambahkan kejadian eksekusi ke diagram untuk menunjukkan saat setiap aktor atau objek sedang dalam proses mengeksekusi salah satu operasinya. Selanjutnya, kita harus memvalidasi diagram. Akhirnya, kita harus mereplikasi proses ini untuk delapan skenario lainnya.



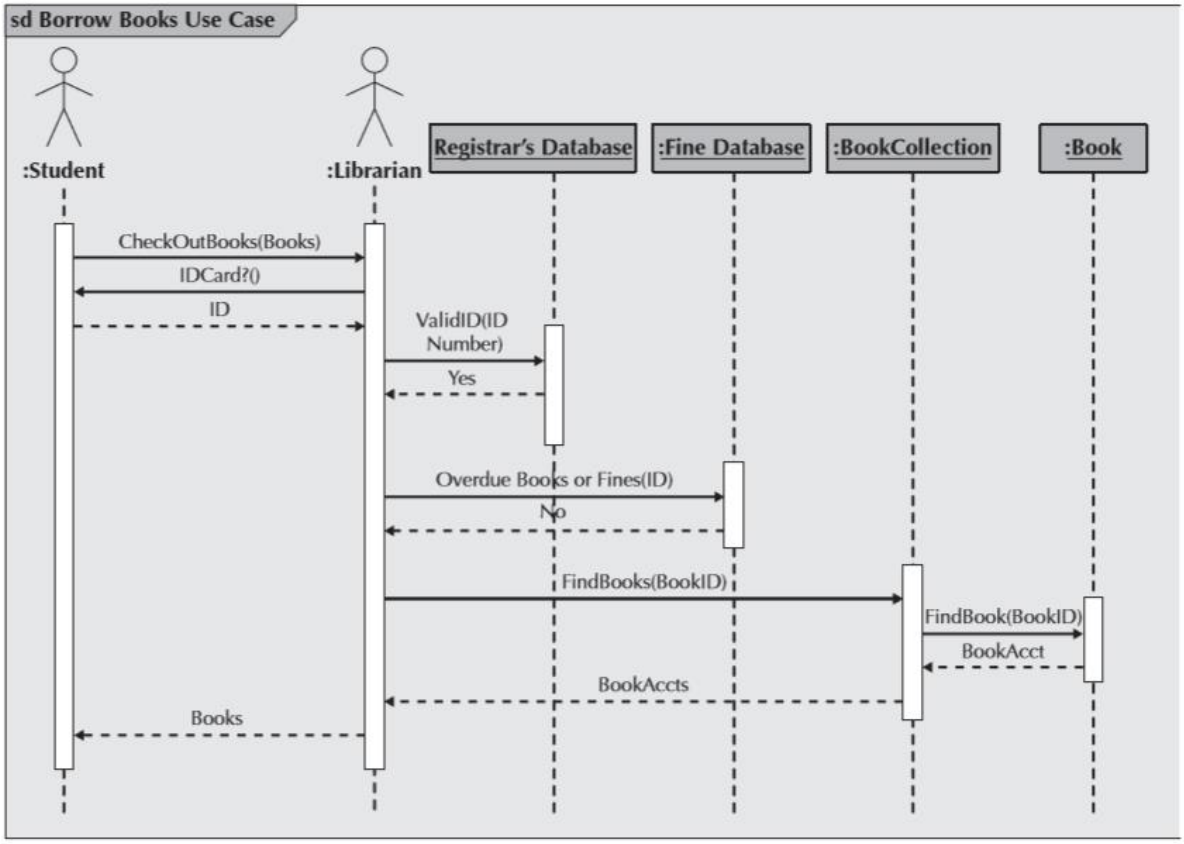
Gambar 6-8 Use-Case Diagram Sistem Pengelolaan Koleksi Buku Perpustakaan (Gambar 4-6)

Diagram Komunikasi

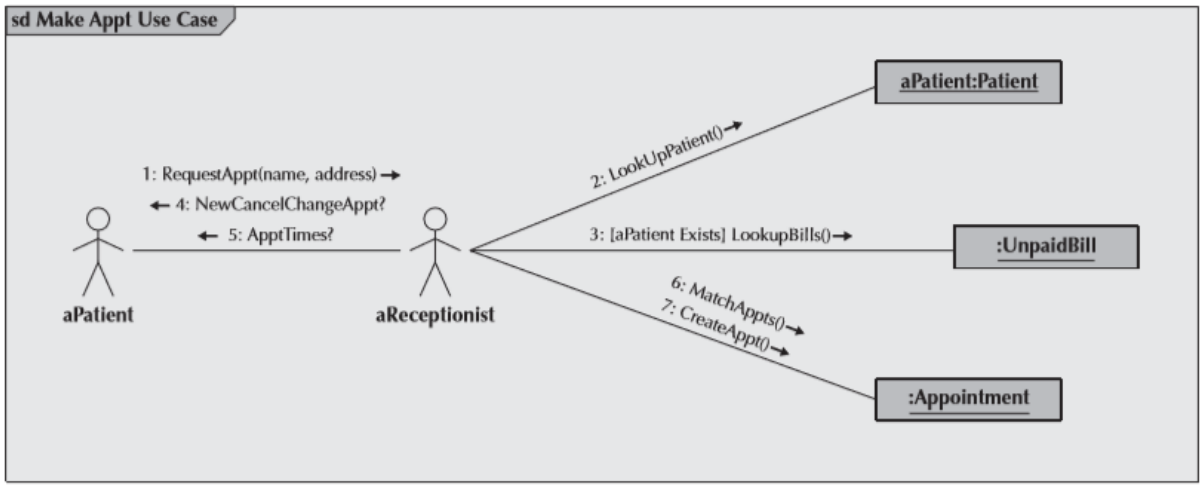
Diagram komunikasi, seperti diagram urutan, pada dasarnya memberikan pandangan tentang aspek dinamis dari sistem berorientasi objek. Mereka dapat menunjukkan bagaimana anggota dari sekumpulan objek berkolaborasi untuk mengimplementasikan use case atau skenario use case. Mereka juga dapat digunakan untuk memodelkan semua interaksi di antara sekumpulan objek yang berkolaborasi, dengan kata lain, sebuah kolaborasi (lihat kartu CRC di Bab 5). Dalam hal ini, diagram komunikasi dapat menggambarkan bagaimana ketergantungan objek yang berbeda satu sama lain. Diagram komunikasi pada dasarnya adalah diagram objek yang menunjukkan hubungan penyampaian pesan alih-alih asosiasi agregasi atau generalisasi. Diagram komunikasi sangat berguna untuk menunjukkan pola proses (yaitu, pola aktivitas yang terjadi pada sekumpulan kelas yang berkolaborasi).

Diagram komunikasi setara dengan diagram urutan, tetapi mereka menekankan aliran pesan melalui satu set objek, sedangkan diagram urutan fokus pada urutan waktu pesan yang dikirimkan. Oleh karena itu, untuk memahami aliran kontrol atas sekumpulan objek yang berkolaborasi atau untuk memahami objek mana yang berkolaborasi untuk mendukung proses bisnis, diagram komunikasi dapat digunakan. Untuk urutan waktu pesan, diagram urutan harus digunakan. Dalam beberapa kasus, keduanya dapat digunakan untuk lebih memahami aktivitas dinamis sistem.

Elemen Diagram Komunikasi Gambar 6-10 menunjukkan diagram komunikasi untuk use case Make Old Patient Appt. Seperti diagram urutan pada Gambar 6-1, proses Make Old Patient Appt digambarkan.

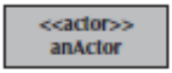
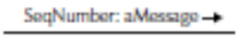
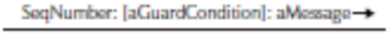


Gambar 6-9 Sequence Diagram Use Case Peminjaman Buku Bagi Siswa BerKTP Yang Masih Berlaku dan Tidak Ada Keterlambatan Buku atau Denda



Gambar 6-10 Contoh Diagram Komunikasi

Aktor dan objek yang berkolaborasi untuk mengeksekusi use case ditempatkan pada diagram komunikasi dengan cara untuk menekankan penyampaian pesan yang terjadi di antara mereka. Perhatikan bahwa aktor dan objek pada Gambar 6-10 adalah sama pada Gambar 6-1: aPatient, aReceptionist, aPatient, UnpaidBill, dan Appointment. Sekali lagi, seperti diagram urutan, untuk setiap objek, nama kelas di mana mereka adalah sebuah instance diberikan setelah nama objek (misalnya, aPatient: Patient). (Sintaks diagram komunikasi diberikan pada Gambar 6-11.) Tidak seperti diagram urutan, diagram komunikasi tidak memiliki sarana untuk secara eksplisit menunjukkan objek yang dihapus atau dibuat. Diasumsikan bahwa ketika pesan hapus, hancurkan, atau hapus dikirim ke suatu objek, itu akan hilang, dan pesan buat atau pesan baru akan menyebabkan objek baru muncul. Perbedaan lain antara dua diagram interaksi adalah bahwa diagram komunikasi tidak pernah menunjukkan pengembalian dari pengiriman pesan, sedangkan diagram urutan dapat menunjukkannya secara opsional.

Istilah dan Definisi	Simbol
Aktor: ■ Adalah orang atau sistem yang mendapatkan peran di luar sistem. ■ Berpartisipasi dalam kolaborasi dengan mengirim dan/atau menerima pesan. ■ Digambarkan sebagai figur tongkat (default) atau, jika aktor nonmanusia terlibat, digambarkan persegi panjang dengan <<actor>> di dalamnya (alternatif).	 anActor 
Objek: ■ Berpartisipasi dalam kolaborasi dengan mengirim dan/atau menerima pesan	
Asosiasi: ■ Menunjukkan hubungan antara aktor dan/atau objek. ■ Digunakan untuk mengirim pesan.	
Pesan: ■ Menyampaikan informasi dari satu objek ke objek lainnya. ■ Memiliki arah, ditunjukkan menggunakan panah. ■ Memiliki urutan, ditunjukkan dengan nomor urut.	
Guard condition: ■ Merupakan tes yang harus dipenuhi untuk pesan yang akan dikirim.	
Bingkai: ■ Menunjukkan konteks diagram komunikasi	

Gambar 6-11 Sintaks Diagram Komunikasi

Asosiasi ditunjukkan antara aktor dan objek dengan garis tidak berarah. Misalnya, asosiasi ditampilkan antara aktor aPatient dan aReceptionist. Pesan ditampilkan sebagai label pada asosiasi. Termasuk dengan label adalah garis dengan panah yang menunjukkan arah pesan yang dikirim. Misalnya, pada Gambar 6-10, aktor aPatient mengirim pesan RequestAppt() ke aktor aReceptionist, dan aktor aReceptionist mengirim pesan

NewCancelChangeAppt?() dan ApptTimes?() ke aktor aPatient. Urutan pengiriman pesan ditandai dengan nomor urut. Pada Gambar 6-10, pesan RequestAppt() adalah pesan pertama yang dikirim, sedangkan pesan NewCancelChangeAppt?() dan ApptTimes?() masing-masing adalah pesan keempat dan kelima yang dikirim.

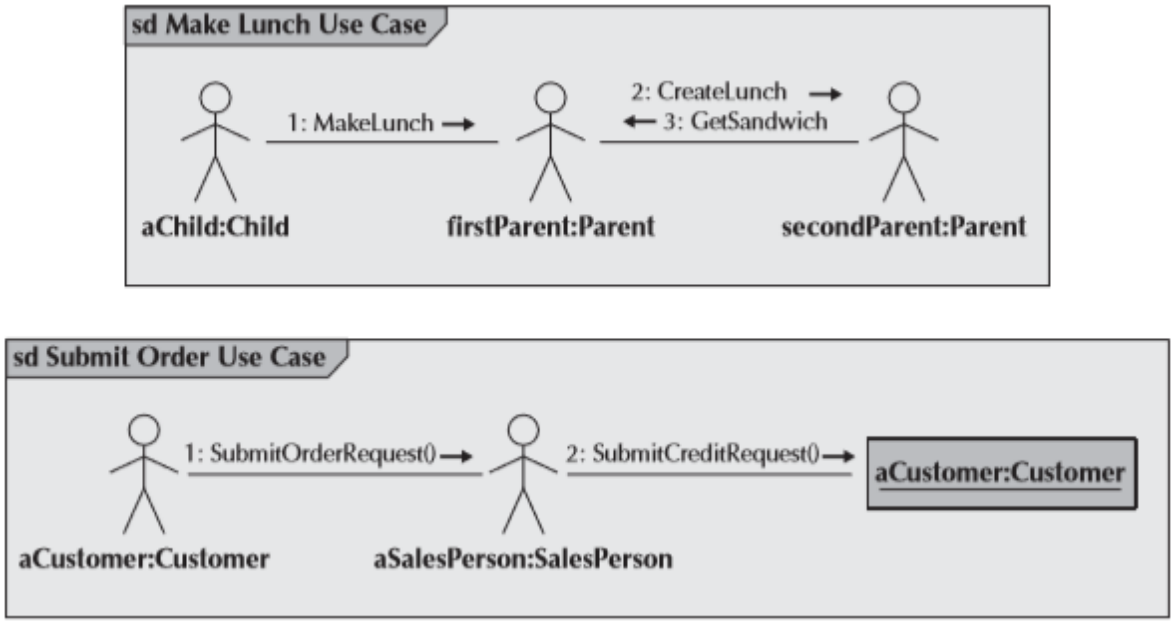
Seperti diagram urutan, diagram komunikasi dapat mewakili pesan bersyarat. Misalnya, pada Gambar 6-10, pesan LookupBills() dikirim hanya jika kondisi [aPatient existing] terpenuhi. Jika pesan dikirim berulang kali, tanda bintang ditempatkan setelah nomor urut. Akhirnya, sebuah asosiasi yang loop ke objek menunjukkan pendelegasian diri. Pesan ditampilkan sebagai label asosiasi.

Ketika diagram komunikasi terisi penuh dengan semua objek, itu bisa menjadi sangat kompleks dan sulit untuk dipahami. Ketika ini terjadi, perlu untuk menyederhanakan diagram. Salah satu pendekatan untuk menyederhanakan diagram komunikasi, seperti diagram use-case (lihat Bab 4) dan diagram kelas (lihat Bab 5), adalah melalui penggunaan paket (yaitu, kelompok logis dari kelas). Dalam kasus diagram komunikasi, objeknya dikelompokkan bersama berdasarkan pesan yang dikirim dan diterima dari objek lain.

Gambar 6-12 memberikan dua contoh tambahan diagram komunikasi. Diagram ini setara dengan diagram urutan yang terdapat pada Gambar 6-3. Namun, ketika membandingkan diagram komunikasi dengan diagram urutan dalam gambar-gambar ini, Anda melihat bahwa cukup banyak informasi yang hilang. Misalnya, pesan CreateSandwich() tidak ditemukan di mana pun. Namun, tujuan utama dari diagram komunikasi adalah untuk menunjukkan bagaimana aktor dan kelas yang berbeda berinteraksi, dan inilah informasi yang disertakan.

Panduan untuk Membuat Diagram Komunikasi, Ambler menyediakan seperangkat panduan saat menggambar diagram komunikasi. Di bagian ini, selain empat pedoman pertama untuk menggambar diagram urutan, kami mempertimbangkan dua lagi.

- Gunakan diagram yang benar untuk informasi yang ingin Anda komunikasikan dengan pengguna. Diagram komunikasi memungkinkan tim untuk dengan mudah mengidentifikasi satu set objek yang saling terkait. Jangan gunakan diagram komunikasi untuk memodelkan aliran proses. Sebagai gantinya, Anda harus menggunakan diagram aktivitas dengan swimlanes yang mewakili objek (lihat Bab 4). Di sisi lain, akan sangat sulit untuk "melihat" bagaimana objek-objek tersebut berkolaborasi dalam diagram aktivitas.
- Saat mencoba memahami urutan pesan, diagram urutan harus digunakan sebagai pengganti diagram komunikasi. Seperti pada pedoman sebelumnya, pedoman ini pada dasarnya menyarankan bahwa Anda harus menggunakan diagram yang dirancang untuk menangani masalah yang dihadapi. Meskipun diagram komunikasi dapat menunjukkan urutan pesan, ini tidak pernah dimaksudkan untuk menjadi tujuan utama mereka.



Gambar 6-12 Contoh Diagram Komunikasi Tambahan

Membuat Diagram Komunikasi. Ingatlah bahwa diagram komunikasi pada dasarnya adalah diagram objek yang menunjukkan hubungan penyampaian pesan alih-alih asosiasi agregasi atau generalisasi. Pada bagian ini, kami menjelaskan proses lima langkah yang digunakan untuk membangun diagram komunikasi. Langkah pertama dalam proses ini adalah menentukan konteks diagram komunikasi. Seperti diagram urutan, konteks diagram dapat berupa sistem, use case, atau skenario use case. Konteks diagram digambarkan sebagai bingkai berlabel di sekitar diagram (lihat Gambar 6-10, 6-11, dan 6-12).

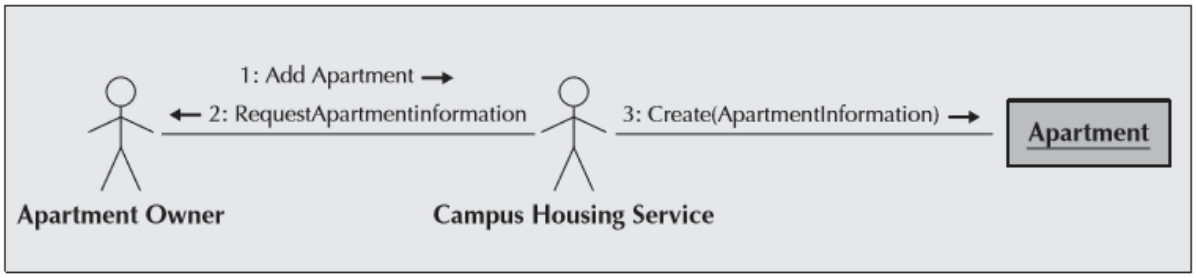
Langkah kedua adalah mengidentifikasi objek (aktor) dan asosiasi yang menghubungkan objek (aktor) yang berpartisipasi dalam kolaborasi bersama. Ingat, objek yang berpartisipasi dalam kolaborasi adalah instance dari kelas yang diidentifikasi selama pengembangan model struktural (lihat Bab 5). Seperti proses diagram urutan, kemungkinan objek tambahan, dan karenanya kelas, akan ditemukan. Sekali lagi, ini normal karena proses pengembangan yang mendasarinya berulang dan bertahap. Selain diagram komunikasi yang dimodifikasi, diagram urutan dan model struktural mungkin juga harus dimodifikasi. Persyaratan fungsional tambahan mungkin juga ditemukan, sehingga memerlukan model fungsional untuk dimodifikasi juga (lihat Bab 4).

Langkah ketiga adalah meletakkan objek (aktor) dan asosiasinya pada diagram komunikasi dengan menempatkannya bersama-sama berdasarkan asosiasi yang mereka miliki dengan objek lain dalam kolaborasi. Dengan berfokus pada asosiasi antara objek (aktor) dan meminimalkan jumlah asosiasi yang saling bersilangan, kita dapat meningkatkan pemahaman diagram.

Langkah keempat adalah menambahkan pesan ke asosiasi antara objek. Kami melakukan ini dengan menambahkan nama pesan ke tautan asosiasi antara objek dan panah yang menunjukkan arah pesan yang dikirim. Setiap pesan memiliki nomor urut yang terkait dengannya untuk menggambarkan urutan pesan berdasarkan waktu.

Langkah kelima dan terakhir adalah memvalidasi diagram komunikasi. Tujuan dari langkah ini adalah untuk menjamin bahwa diagram komunikasi dengan tepat menggambarkan proses yang mendasarinya. Hal ini dilakukan dengan memastikan bahwa semua langkah dalam proses digambarkan pada diagram.

C**ontoh Perumahan Kampus.** Seperti contoh sequence diagram, kita kembali ke use case Tambah Apartemen untuk Layanan Perumahan Kampus. Untuk memulainya, kami kembali mengatur konteks untuk diagram komunikasi (Use-Case Tambahkan Apartemen). Selanjutnya dilakukan identifikasi objek (Apartemen), aktor (Pemilik Apartemen dan Dinas Perumahan Kampus), dan asosiasi (keterkaitan antara aktor Pemilik Apartemen dengan aktor Dinas Perumahan Kampus dan keterkaitan antara aktor Dinas Perumahan Kampus dan objek Apartemen). Dengan menggunakan informasi ini, kami menyusun diagram yang menunjukkan aktor, objek, dan asosiasi di antara mereka. Akhirnya, kami memberi label asosiasi dengan pesan yang sesuai. Gambar 6-13 menggambarkan diagram komunikasi untuk use case ini.



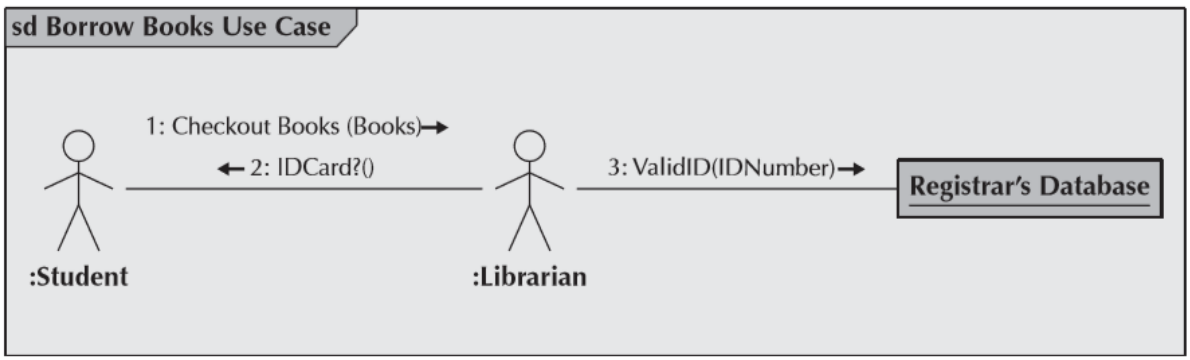
Gambar 6-13 Diagram Komunikasi untuk Add Apartment Use Case

Contoh Perpustakaan. Seperti contoh diagram urutan, kita kembali ke Use-Case Pinjam Buku dari Sistem Manajemen Koleksi Buku Perpustakaan. Dalam hal ini, untuk mengatur konteks diagram, kami mengunjungi Siswa tanpa ID yang valid dan Siswa dengan ID yang Valid tetapi memiliki skenario denda atau memiliki buku yang terlambat. Kami membuat dua diagram komunikasi, satu untuk setiap skenario. Seperti pada proses sequence-diagram, kami meninjau Flow of Events dari deskripsi use-case (lihat Gambar 6-6), diagram aktivitas (lihat Gambar 6-7), dan diagram use case (lihat Gambar 6-8).

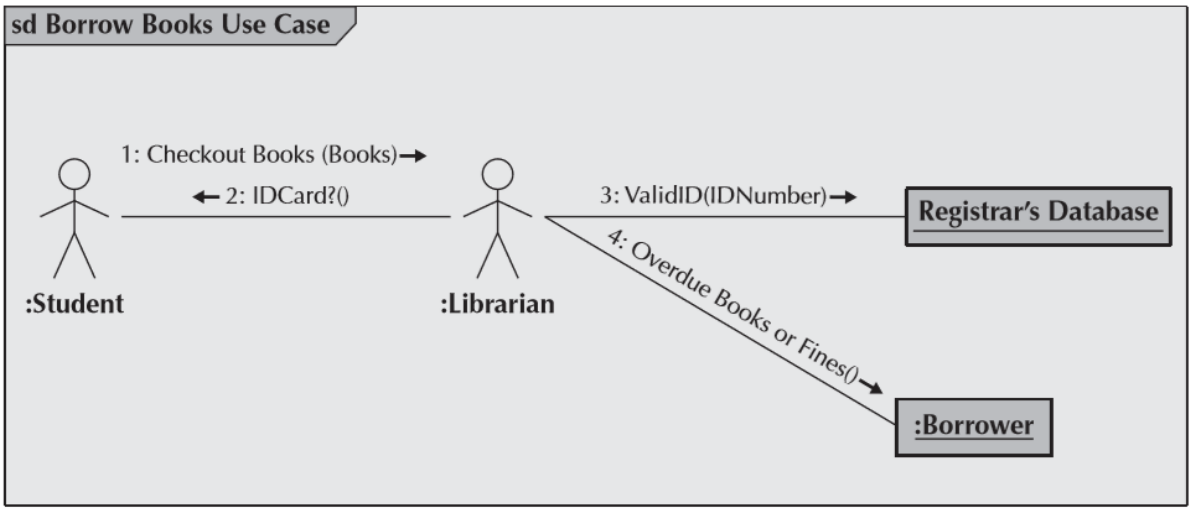
Langkah selanjutnya adalah mengidentifikasi aktor, objek, dan asosiasi yang terlibat dalam skenario. Dalam kedua skenario, aktornya adalah Siswa, Pustakawan, dan Database Panitera. Namun, karena proses dibatalkan sangat awal di Siswa tanpa skenario ID yang valid, tidak ada objek dalam skenario. Pelajar dengan KTP yang masih berlaku tetapi berutang denda atau memiliki buku yang terlambat skenario termasuk satu objek: Peminjam. Kedua skenario memiliki asosiasi antara aktor Pelajar dan Pustakawan dan aktor Database Pustakawan dan Panitera. Pelajar dengan KTP yang masih berlaku tetapi berhutang denda atau memiliki skenario buku yang terlambat juga memiliki hubungan antara aktor Pustakawan dan objek Peminjam.

Langkah selanjutnya adalah membuat diagram. Dalam kedua kasus, karena siswa memulai proses, kami menempatkan aktor Siswa di paling kiri diagram. Kami kemudian menempatkan aktor lain pada diagram dalam urutan di mana mereka berpartisipasi dalam proses. Kami juga menempatkan objek :Peminjam ke paling kanan bawah diagram yang mewakili Siswa dengan ID yang Valid tetapi berhutang denda atau memiliki skenario buku yang terlambat untuk mencerminkan arah membaca kiri-ke-kanan dan atas-ke-bawah untuk sebagian besar budaya Barat.

Sekarang kita menempatkan asosiasi yang relevan antara aktor dan objek yang berpartisipasi dalam skenario. Pada langkah ini, kami menambahkan pesan ke asosiasi. Kami kembali meninjau aliran peristiwa (lihat Gambar 6-6) dari deskripsi Use-Case untuk mengidentifikasi arah dan isi pesan. Gambar 6-14 dan 6-15 menggambarkan diagram komunikasi yang dibuat.



Gambar 6-14 Diagram Komunikasi Siswa Tanpa Kartu Identitas yang Sah



Gambar 6-15 Diagram Komunikasi Siswa Ber KTP yang Masih Berlaku Namun Terutang Denda atau Memiliki Keterlambatan Buku

Langkah terakhir adalah memvalidasi diagram. Seperti diagram urutan, karena kita menggambar contoh versi spesifik dari diagram komunikasi, kita juga harus menggambar tujuh diagram yang tersisa untuk skenario lainnya.

6.5 MESIN STATUS PERILAKU

Beberapa kelas dalam diagram kelas mewakili satu set objek yang cukup dinamis karena mereka melewati berbagai keadaan selama keberadaannya. Misalnya, pasien dapat berubah dari waktu ke waktu dari yang baru menjadi yang sekarang menjadi yang lama berdasarkan statusnya di kantor dokter. Mesin status perilaku adalah model dinamis yang menunjukkan status berbeda yang dilalui satu objek selama hidupnya sebagai respons terhadap peristiwa, bersama dengan respons dan tindakannya. Biasanya, mesin status perilaku tidak digunakan untuk semua objek; alih-alih, mesin status perilaku digunakan dengan objek kompleks untuk mendefinisikannya lebih lanjut dan membantu menyederhanakan desain algoritme untuk metodenya. Mesin status perilaku menunjukkan status objek yang berbeda dan peristiwa apa yang menyebabkan objek berubah dari satu status ke status lainnya. Mesin status perilaku harus digunakan untuk membantu memahami aspek dinamis dari satu kelas dan bagaimana instance-nya berkembang dari waktu ke waktu15 tidak seperti diagram interaksi yang menunjukkan bagaimana Use-Case atau skenario Use-Case tertentu dieksekusi pada sekumpulan kelas.

Di bagian ini, kami menjelaskan status, peristiwa, transisi, tindakan, dan aktivitas. Kami juga menjelaskan bagaimana mesin status perilaku memodelkan perubahan status yang dilalui objek kompleks. Seperti diagram interaksi, ketika kita membuat mesin status perilaku untuk suatu objek, ada kemungkinan bahwa kita akan mengungkap peristiwa tambahan yang perlu dimasukkan dalam model fungsional (lihat Bab 4) dan operasi tambahan yang perlu dimasukkan dalam struktur. model (lihat Bab 5), jadi diagram interaksi kita mungkin harus dimodifikasi lagi. Karena pengembangan berorientasi objek bersifat iteratif dan inkremental, modifikasi berkelanjutan dari model yang berkembang (fungsional, struktural, dan perilaku) dari sistem ini diharapkan.

Status, Peristiwa, Transisi, Tindakan, dan Aktivitas

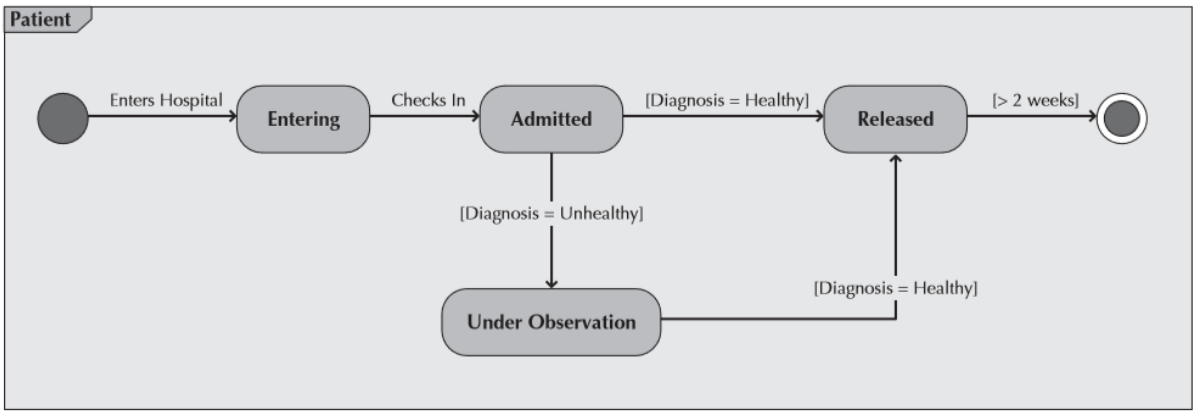
Keadaan suatu objek ditentukan oleh nilai atributnya dan hubungannya dengan objek lain pada titik waktu tertentu. Misalnya, pasien mungkin memiliki keadaan baru, saat ini, atau mantan. Atribut atau properti dari suatu objek mempengaruhi keadaan di mana ia berada; namun, tidak semua atribut atau perubahan atribut akan membuat perbedaan. Misalnya, pikirkan tentang alamat pasien. Atribut-atribut tersebut membuat perbedaan yang sangat kecil terhadap perubahan keadaan pasien. Namun, jika negara bagian didasarkan pada lokasi geografis pasien (misalnya, pasien dalam kota diperlakukan berbeda dari pasien luar kota), perubahan alamat pasien akan memengaruhi perubahan status.

Peristiwa adalah sesuatu yang terjadi pada titik waktu tertentu dan mengubah nilai atau nilai yang menggambarkan suatu objek, yang pada gilirannya mengubah keadaan objek tersebut. Ini bisa berupa kondisi yang ditentukan menjadi benar, penerimaan panggilan untuk metode oleh objek, atau berlalunya periode waktu yang ditentukan. Keadaan objek menentukan dengan tepat apa responsnya.

Transisi adalah hubungan yang merepresentasikan perpindahan suatu objek dari satu keadaan ke keadaan lain. Beberapa transisi memiliki kondisi penjaga. Kondisi penjaga adalah ekspresi Boolean yang menyertakan nilai atribut, yang memungkinkan transisi terjadi hanya jika kondisinya benar. Sebuah objek biasanya bergerak dari satu keadaan ke keadaan lain berdasarkan hasil dari suatu tindakan yang dipicu oleh suatu peristiwa. Tindakan adalah proses atomik yang tidak dapat didekomposisi yang tidak dapat diinterupsi. Dari perspektif praktis, tindakan membutuhkan waktu nol, dan mereka terkait dengan transisi. Sebaliknya, suatu aktivitas adalah proses nonatomik yang dapat diurai yang dapat diinterupsi. Kegiatan membutuhkan waktu yang lama untuk diselesaikan, dan dapat dimulai dan dihentikan oleh suatu tindakan.

Elemen dari Behavioral State Machine

Gambar 6-16 menyajikan contoh mesin status perilaku yang mewakili kelas pasien dalam konteks lingkungan rumah sakit. Dari diagram ini, kita dapat mengetahui bahwa seorang pasien masuk rumah sakit dan dirawat setelah check in. Jika seorang dokter menemukan pasien itu sehat, ia akan dibebaskan dan tidak lagi dianggap sebagai pasien setelah dua minggu berlalu. Jika seorang pasien ditemukan tidak sehat, ia tetap di bawah pengawasan sampai diagnosis berubah.



Gambar 6-16 Contoh Diagram Behavioral State Machine

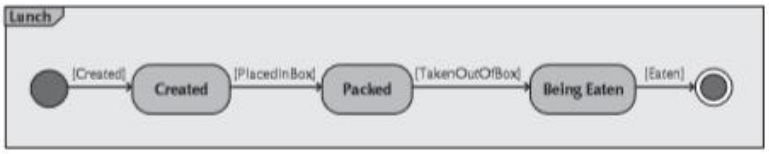
State adalah sekumpulan nilai yang menggambarkan suatu objek pada suatu titik waktu tertentu dan mewakili suatu titik dalam kehidupan suatu objek di mana objek tersebut memenuhi beberapa kondisi, melakukan beberapa tindakan, atau menunggu sesuatu terjadi (lihat Gambar 6-17). Pada Gambar 6-16 keadaan termasuk masuk, diterima, dilepaskan, dan dalam pengamatan. Sebuah negara digambarkan oleh simbol negara, yang merupakan persegi panjang dengan sudut membulat dengan label deskriptif yang mengkomunikasikan keadaan tertentu. Ada dua pengecualian. Keadaan awal ditunjukkan menggunakan lingkaran kecil yang terisi, dan keadaan akhir objek ditampilkan sebagai lingkaran yang mengelilingi lingkaran kecil yang terisi. Pengecualian ini menggambarkan kapan suatu objek mulai dan tidak ada lagi, masing-masing.

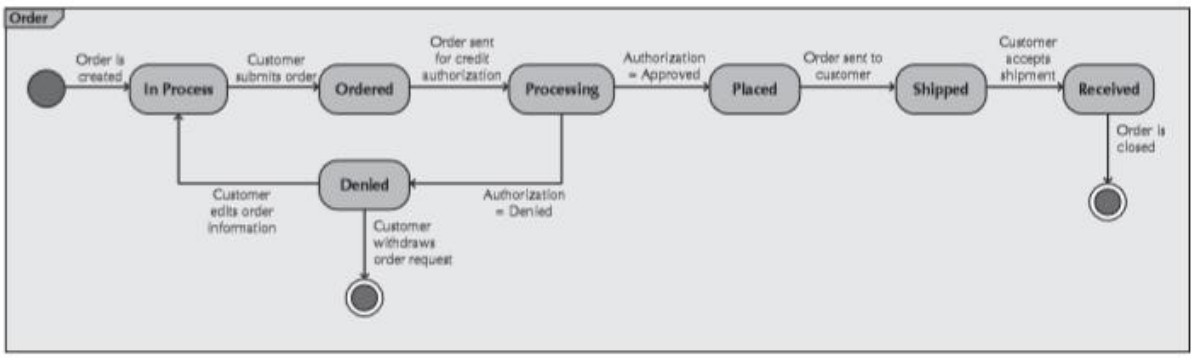
Panah digunakan untuk menghubungkan simbol negara, mewakili transisi antar negara. Setiap panah diberi label dengan nama peristiwa yang sesuai dan parameter atau ketentuan apa pun yang mungkin berlaku. Misalnya, dua transisi dari diterima ke dibebaskan dan di bawah pengawasan mengandung kondisi penjagaan. Seperti dalam diagram perilaku lainnya, dalam banyak kasus berguna untuk secara eksplisit menunjukkan konteks mesin status perilaku menggunakan bingkai.

Gambar 6-18 menggambarkan dua mesin status perilaku tambahan. Yang pertama adalah untuk objek makan siang yang diasosiasikan dengan skenario use case Make Lunch dari Gambar 6-3 dan 6-12. Dalam hal ini, jelas ada informasi tambahan yang telah ditangkap tentang objek makan siang. Misalnya, skenario Gambar 6-3 dan 6-12 tidak memasukkan informasi mengenai makan siang yang dikeluarkan dari kotak atau dimakan. Ini menyiratkan Use-Case tambahan dan/atau skenario Use-Case yang harus disertakan dalam sistem yang menangani pemrosesan makan siang. Mesin status perilaku kedua berhubungan dengan siklus hidup suatu pesanan. Objek pesanan dikaitkan dengan skenario Use-Case pesanan yang dijelaskan dalam Gambar 6-3 dan 6-12. Seperti pada contoh makan siang, ada sedikit informasi tambahan yang terkandung dalam mesin status perilaku ini. Untuk sistem pemrosesan pesanan, diagram urutan dan komunikasi tambahan akan diperlukan untuk sepenuhnya mewakili semua pemrosesan yang terkait dengan objek pesanan. Jelas, karena mesin status perilaku dapat mengungkap persyaratan pemrosesan tambahan, mereka bisa sangat berguna dalam mengisi deskripsi lengkap dari sistem yang berkembang.

Istilah dan Definisi	Simbol
Keadaan ■ Ditampilkan dengan bentuk persegi panjang dengan sudut membulat. ■ Memiliki nama yang mewakili keadaan suatu objek.	aState
Kondisi awal: ■ Ditampilkan dengan bentuk lingkaran kecil yang terisi. ■ Merupakan titik di mana suatu objek mulai ada.	
Kondisi akhir: ■ Ditampilkan dengan bentuk lingkaran yang mengelilingi lingkaran kecil yang terisi (bulll's-eye). ■ Melambangkan selesainya kegiatan.	
Peristiwa: ■ Adalah kejadian penting yang memicu perubahan keadaan. ■ Dapat berupa kondisi yang ditentukan menjadi benar, penerimaan sinyal eksplisit dari satu objek ke objek lain, atau berlalunya periode waktu yang ditentukan. ■ Digunakan untuk menandai transisi.	anEvent
Transisi: ■ Menunjukkan bahwa suatu objek dalam keadaan pertama akan memasuki keadaan kedua. ■ Dipicu oleh terjadinya peristiwa pelabelan transisi. ■ Ditampilkan sebagai panah padat dari satu keadaan ke keadaan lain, diberi label dengan nama peristiwa.	
Bingkai: ■ Menunjukkan konteks mesin status perilaku.	Context

Gambar 6-17 Sintaks Diagram Mesin Status Perilaku

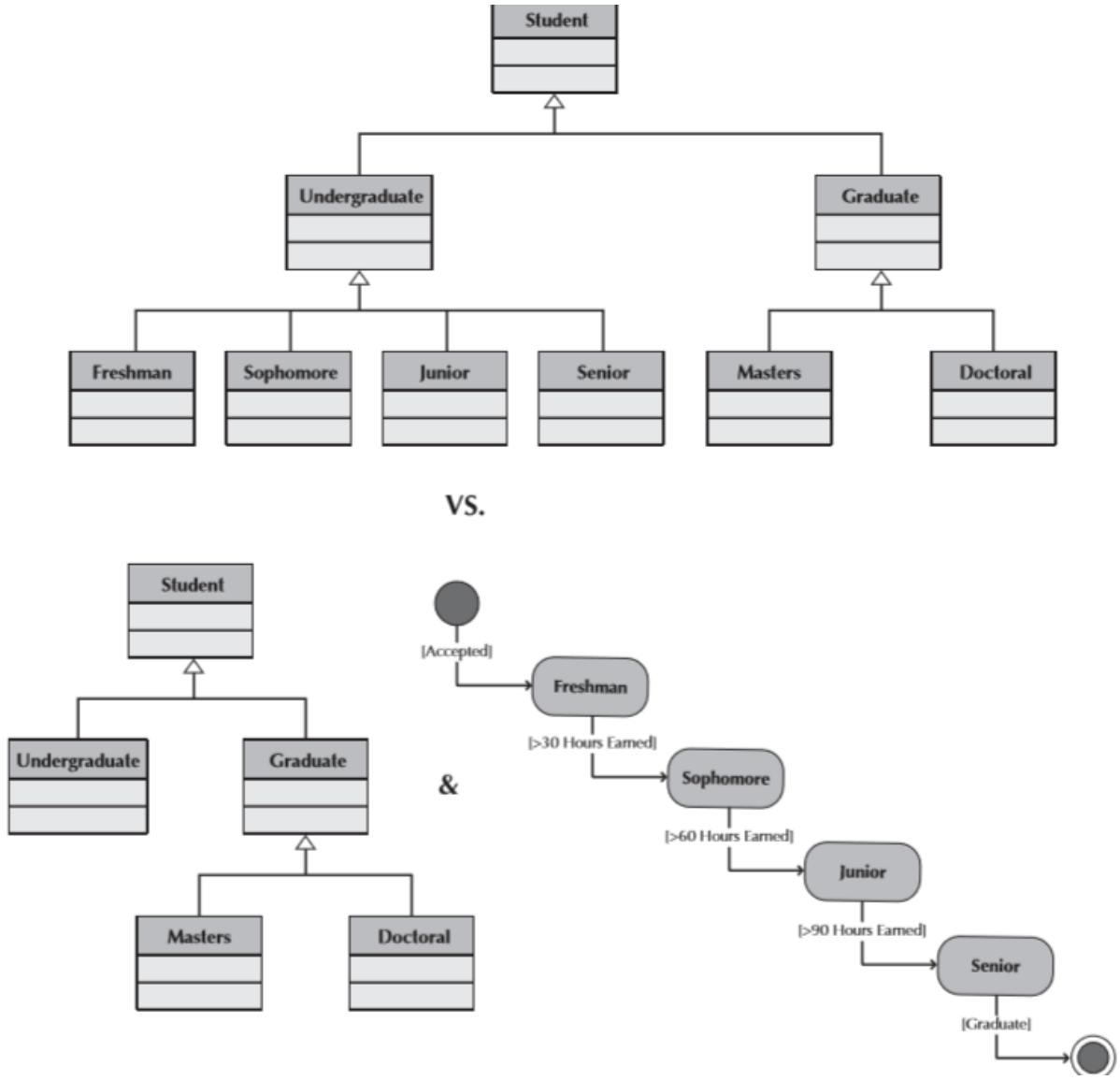




Gambar 6-18 Diagram Mesin Status Perilaku Tambahan

Terkadang, state dan subclass bisa dikacaukan. Misalnya, pada Gambar 6-19, apakah kelas-kelas Freshman, Sophomore, Junior, dan Senior merupakan subclass dari kelas Undergraduate atau apakah mereka menyatakan bahwa sebuah instance dari kelas Undergraduate dilalui selama masa hidupnya? Dalam hal ini, yang terakhir adalah jawaban yang lebih baik. Saat mencoba mengidentifikasi semua kelas potensial selama pemodelan struktural (lihat Bab 5), Anda mungkin benar-benar mengidentifikasi status superkelas yang

relevan alih-alih subkelas. Ini adalah contoh lain betapa eratnya model fungsional, struktural, dan perilaku dapat terjalin. Dari perspektif pemodelan, meskipun kami akhirnya menghapus subkelas Freshman, Sophomore, Junior, dan Senior dari model struktural, menangkap informasi itu selama pemodelan struktural dan menghapusnya berdasarkan penemuan yang dibuat selama pemodelan perilaku lebih disukai daripada menghilangkannya dan mengambil kesempatan kehilangan informasi penting tentang domain masalah. Ingat, pengembangan berorientasi objek bersifat iteratif dan inkremental. Saat kita maju ke model domain masalah yang benar, kita akan membuat banyak kesalahan.



Gambar 6-19 Negara versus Subkelas

- Panduan untuk Membuat Mesin Status Perilaku Seperti halnya diagram urutan dan komunikasi, Amble menyarankan seperangkat panduan saat menggambar mesin status perilaku. Dalam hal ini, kami mempertimbangkan enam rekomendasinya.
- Buat mesin status perilaku untuk objek yang perilakunya berubah berdasarkan status objek. Dengan kata lain, jangan membuat mesin status perilaku untuk objek yang perilakunya selalu sama terlepas dari statusnya. Objek-objek ini terlalu sederhana.
- Untuk mematuhi konvensi pembacaan kiri-ke-kanan dan atas-ke-bawah budaya Barat, keadaan awal harus digambar di sudut kiri atas diagram dan keadaan akhir harus digambar di kanan bawah diagram.

- Pastikan nama-nama negara bagian sederhana, jelas secara intuitif, dan deskriptif. Misalnya pada Gambar 6-16, nama state objek pasien adalah Entering, Admitted, Under Observation, dan Released.
- Pertanyaan lubang hitam dan keadaan keajaiban. Jenis keadaan ini bermasalah karena alasan yang sama lubang hitam dan aktivitas keajaiban adalah masalah untuk diagram aktivitas (lihat Bab 4). Keadaan lubang hitam, menyatakan bahwa suatu objek masuk dan tidak pernah keluar, kemungkinan besar sebenarnya adalah keadaan akhir. Keajaiban menyatakan, menyatakan bahwa suatu objek keluar dari tetapi tidak pernah masuk, kemungkinan besar adalah keadaan awal.
- Pastikan bahwa semua kondisi penjaga saling eksklusif (tidak tumpang tindih). Misalnya, pada Gambar 6-16, kondisi penjaga [Diagnosis = Sehat] dan kondisi penjaga [Diagnosis = Tidak Sehat] tidak tumpang tindih. Namun, jika Anda membuat kondisi penjaga [x >= 0] dan kondisi penjaga kedua [x <= 0], kondisi penjaga tumpang tindih ketika x = 0, dan tidak jelas ke status mana objek akan bertransisi. Ini jelas akan menimbulkan kebingungan.
- Semua transisi harus dikaitkan dengan pesan dan operasi. Jika tidak, keadaan objek tidak akan pernah bisa berubah. Meskipun ini mungkin menyatakan yang sudah jelas, ada banyak kali para analis lupa untuk kembali dan memastikan bahwa ini memang benar.

Membuat Mesin Status Perilaku

Mesin status perilaku digambar untuk menggambarkan turunan dari satu kelas dari diagram kelas. Biasanya, kelas sangat dinamis dan kompleks, membutuhkan pemahaman yang baik tentang keadaan mereka dari waktu ke waktu dan peristiwa yang memicu perubahan. Anda harus memeriksa diagram kelas Anda untuk mengidentifikasi kelas mana yang mengalami serangkaian perubahan status yang kompleks dan menggambar diagram untuk masing-masing kelas tersebut. Di bagian ini, kami menjelaskan proses lima langkah yang digunakan untuk membangun mesin status perilaku. Seperti model perilaku lainnya, langkah pertama dalam proses ini adalah menentukan konteks mesin status perilaku, yang ditunjukkan dalam label bingkai diagram. Konteks dari mesin status perilaku biasanya adalah sebuah kelas. Namun, itu juga bisa menjadi satu set kelas, subsistem, atau keseluruhan sistem.

Langkah kedua adalah mengidentifikasi berbagai keadaan yang akan dimiliki suatu objek selama masa hidupnya. Ini termasuk menetapkan batas-batas keberadaan suatu objek dengan mengidentifikasi keadaan awal dan akhir suatu objek. Kita juga harus mengidentifikasi keadaan suatu objek. Informasi yang diperlukan untuk melakukan ini diperoleh dari membaca deskripsi Use-Case, berbicara dengan pengguna, dan mengandalkan teknik pengumpulan persyaratan yang Anda pelajari di Bab 3. Cara mudah untuk mengidentifikasi status suatu objek adalah dengan menulis langkah-langkah apa yang terjadi pada suatu objek dari waktu ke waktu, dari awal hingga akhir, mirip dengan bagaimana aliran normal bagian peristiwa dari deskripsi Use-Case akan dibuat.

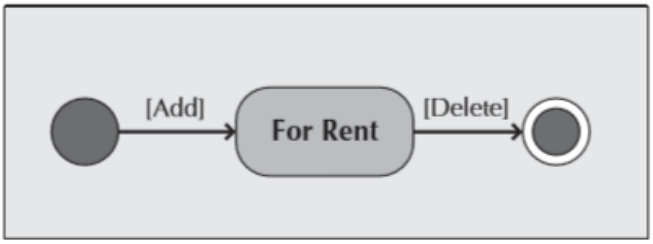
Langkah ketiga adalah menentukan urutan keadaan yang akan dilalui suatu objek selama masa hidupnya. Dengan menggunakan urutan ini, status ditempatkan ke mesin status perilaku dalam urutan kiri-ke-kanan.

Langkah keempat adalah mengidentifikasi transisi antara status objek dan menambahkan peristiwa, tindakan, dan kondisi penjaga yang terkait dengan transisi. Peristiwa adalah pemicu yang menyebabkan suatu objek berpindah dari satu keadaan ke keadaan berikutnya. Dengan kata lain, suatu peristiwa menyebabkan tindakan untuk dieksekusi yang mengubah nilai atribut objek secara signifikan. Tindakan biasanya operasi yang terkandung dalam objek. Juga, kondisi penjaga dapat memodelkan serangkaian kondisi pengujian yang

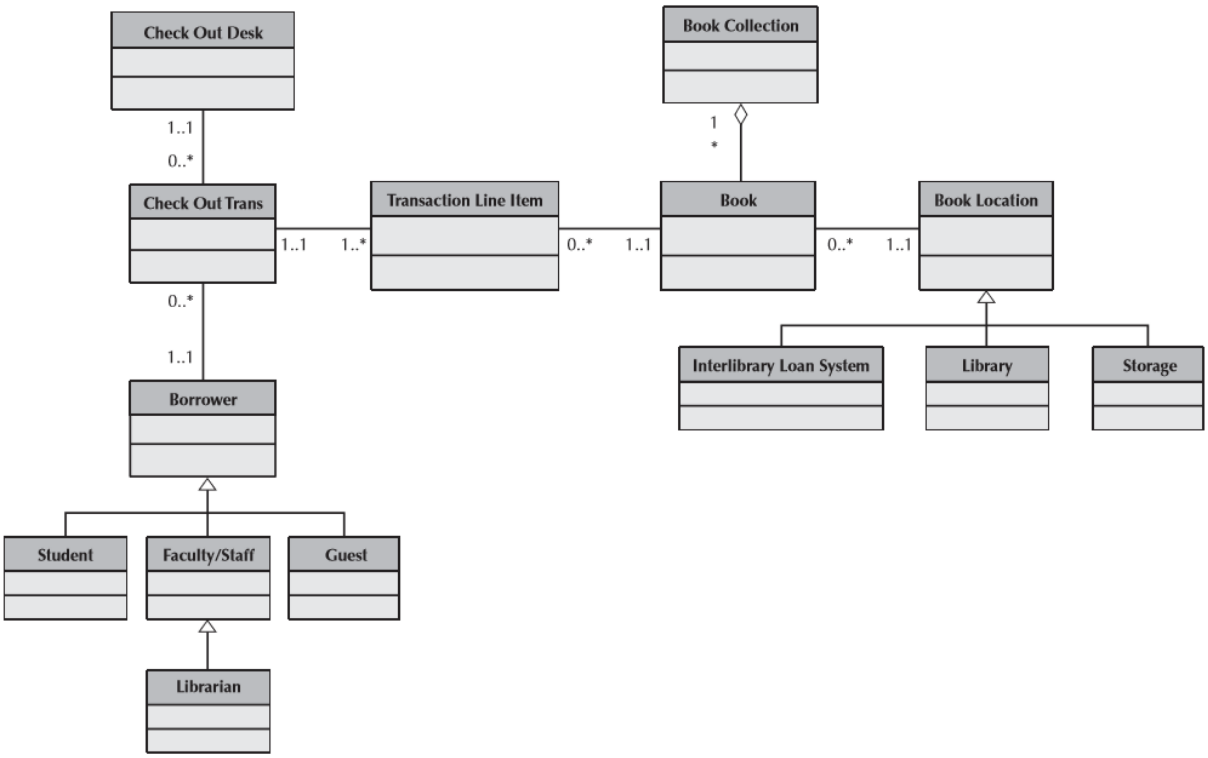
harus dipenuhi agar transisi terjadi. Pada titik ini dalam proses, transisi ditarik antara keadaan yang relevan dan diberi label dengan peristiwa, tindakan, atau kondisi penjaga.

Langkah kelima adalah memvalidasi mesin status perilaku dengan memastikan bahwa setiap status dapat dijangkau dan memungkinkan untuk meninggalkan semua status kecuali untuk status akhir. Jelas, jika status yang diidentifikasi tidak dapat dijangkau, transisi hilang atau status diidentifikasi karena kesalahan. Hanya status akhir yang bisa menjadi jalan buntu dari perspektif siklus hidup objek.

Contoh Perumahan Kampus. Berdasarkan model fungsional dan struktural untuk layanan perumahan kampus (lihat Gambar 6-4), diagram urutan untuk use case Tambah Apartemen (lihat Gambar 6-5), dan diagram komunikasi untuk use case Tambah Apartemen (lihat Gambar 6-13), di bagian ini, kita akan membuat mesin status perilaku untuk kelas Apartemen. Dengan meninjau semua representasi, jelas bahwa mesin status perilaku akan sangat sederhana. Dalam hal ini, sebuah apartemen menjadi ada saat ditambahkan dan menghilang saat dihapus. Satu-satunya negara bagian adalah Disewakan. Gambar 6-20 menggambarkan mesin status perilaku untuk kelas ini.



Gambar 6-20 Behavioral State Machine untuk Kelas Apartemen



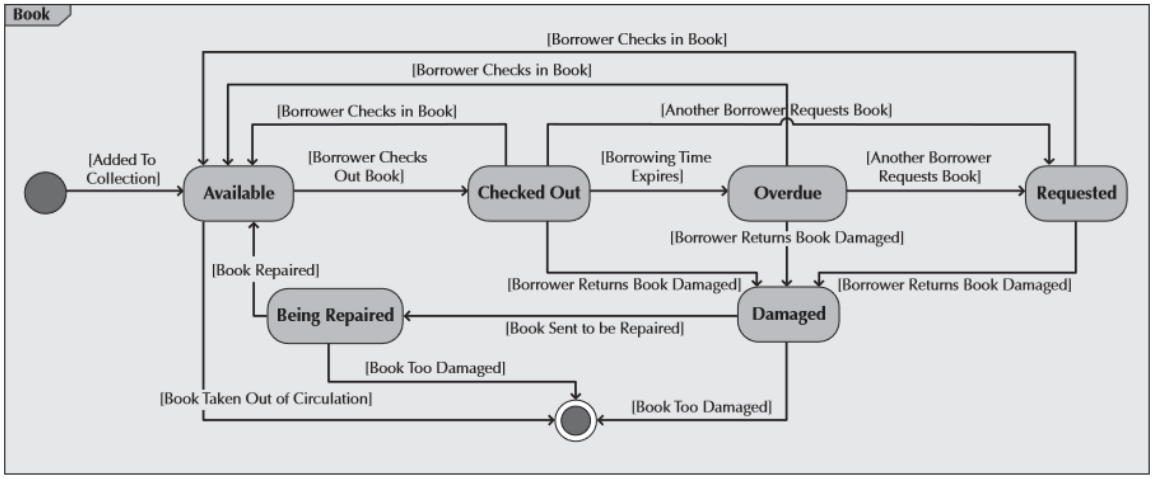
Gambar 6-21 Diagram Kelas Sistem Pengelolaan Koleksi Buku Perpustakaan

Contoh Perpustakaan. Langkah pertama dalam menggambar mesin status perilaku adalah mengatur konteksnya. Untuk tujuan kita, konteks biasanya adalah turunan dari kelas

yang memiliki banyak status dan yang perilakunya bergantung pada status di mana ia berada saat ini. Seperti yang disarankan sebelumnya, kita harus meninjau diagram kelas (lihat Gambar 6-21) untuk mengidentifikasi kelas "menarik". Dalam hal Sistem Manajemen Koleksi Buku Perpustakaan, kelas yang jelas untuk dipertimbangkan adalah kelas Buku.

Langkah selanjutnya adalah mengidentifikasi status berbeda yang dapat dilalui oleh instance kelas Buku selama masa pakainya. Tempat yang baik untuk mencari kemungkinan perubahan keadaan adalah deskripsi Use-Case (lihat Gambar 6-6), diagram aktivitas (lihat Gambar 6-7), diagram urutan (lihat Gambar 6-9), dan diagram komunikasi (lihat Gambar 6-9). Gambar 6-14 dan 6-15). Dalam kasus sebuah buku, meskipun statusnya mungkin serupa, Anda harus berhati-hati dalam mengidentifikasi status yang terkait dengan instance kelas Buku dan bukan status yang terkait dengan buku fisik itu sendiri. Dalam Bab 5, kami mengamati bahwa ada sejumlah keadaan tersirat yang perlu dipertimbangkan. Ini termasuk Check Out, Overdue, Requested, Available, dan Damaged. Jika buku itu rusak, buku itu bisa diperbaiki dan dikembalikan ke peredarannya atau bisa juga terlalu rusak untuk diperbaiki dan dikeluarkan dari peredaran. Meskipun Peminjam dapat didenda karena buku yang jatuh tempo atau rusak, denda bukanlah keadaan buku, itu adalah keadaan peminjam.

Selanjutnya, kami menyusun diagram dengan mengurutkan status secara berurutan berdasarkan siklus hidup sebuah buku. Misalnya, mungkin tidak masuk akal jika sebuah buku berubah dari keadaan diperbaiki ke keadaan rusak. Namun, beralih dari keadaan rusak ke keadaan diperbaiki masuk akal. Juga tidak masuk akal bagi sebuah buku untuk beralih dari keadaan yang tersedia langsung ke keadaan terlambat. Namun, kebalikannya masuk akal. Status yang kami identifikasi untuk objek buku termasuk Tersedia, Diperiksa, Terlambat, Diminta, Rusak, dan Sedang Diperbaiki. Selanjutnya kami menambahkan transisi antara status dan memberi label dengan kondisi penjaga yang sesuai. Mesin status perilaku untuk instance kelas Buku digambarkan pada Gambar 6-22.



Gambar 6-22 Mesin Status Perilaku untuk Instance Kelas Buku dalam Sistem Manajemen Koleksi Buku Perpustakaan

Terakhir, kami memvalidasi diagram dengan memeriksa keadaan atau transisi yang hilang dan memastikan bahwa tidak ada lubang hitam atau keadaan ajaib.

6.6 ANALISIS CRUDE

Salah satu teknik yang berguna untuk mengidentifikasi bagaimana objek yang mendasari dalam domain masalah bekerja sama untuk berkolaborasi dalam mendukung Use-Case adalah analisis CRUDE. Analisis CRUDE menggunakan matriks CRUDE, di mana setiap interaksi antar objek diberi label dengan huruf untuk jenis interaksi: C untuk membuat,

R untuk membaca atau referensi, U untuk pembaruan, D untuk menghapus, dan E untuk mengeksekusi. Dalam pendekatan berorientasi objek, matriks kelas/aktor-oleh-kelas/aktor digunakan. Setiap sel dalam matriks mewakili interaksi antara instance kelas. Sebagai contoh, pada Gambar 6-1, sebuah instance dari aktor Resepsionis membuat sebuah instance dari kelas Appointment. Dengan asumsi Baris:Pemesanan kolom, C ditempatkan di sel Resepsionis: Janji temu. Juga, pada Gambar 6-1, sebuah instance dari aktor Resepsionis mereferensikan sebuah instance dari kelas Appointments. Dalam hal ini, sebuah R ditempatkan di sel Resepsionis: Janji Temu. Gambar 6-23 menunjukkan matriks CRUDE berdasarkan use case Make Old Patient Appt.

Berbeda dengan diagram interaksi dan mesin status perilaku, matriks CRUDE paling berguna sebagai representasi seluruh sistem. Setelah matriks CRUDE selesai untuk seluruh sistem, matriks dapat dipindai dengan cepat untuk memastikan bahwa setiap kelas dapat dipakai. Setiap jenis interaksi dapat divalidasi untuk setiap kelas. Misalnya, jika kelas hanya mewakili objek sementara, maka kolom dalam matriks harus memiliki D di suatu tempat. Jika tidak, instance kelas tidak akan pernah dihapus. Karena gudang data berisi data historis, objek yang akan disimpan dalam satu tidak boleh memiliki entri U atau D di kolom terkait. Dengan cara ini, analisis CRUDE dapat digunakan sebagai cara untuk memvalidasi sebagian interaksi antara objek dalam sistem berorientasi objek. Akhirnya, semakin banyak interaksi di antara satu set kelas, semakin besar kemungkinan mereka harus dikelompokkan bersama dalam sebuah kolaborasi. Namun, jumlah dan jenis interaksi hanya perkiraan pada saat ini dalam pengembangan sistem. Perawatan harus diambil ketika menggunakan teknik ini untuk mengelompokkan kelas untuk mengidentifikasi kolaborasi. Kami kembali ke subjek ini di bab berikutnya ketika kami berurusan dengan partisi dan kolaborasi.

	Receptionist	PatientList	Patient	UnpaidBills	Appointments	Appointment
Receptionist		RU	CRUD	R	RU	CRUD
PatientList			R			
Patient						
UnpaidBills						
Appointments						R
Appointment						

Gambar 6-23 Matriks CRUDE untuk Use-Case Tepat Pasien Lama

Analisis CRUDE juga dapat digunakan untuk mengidentifikasi objek yang kompleks. Semakin banyak entri (C)reate, (U)pdate, atau (D)elete di kolom yang terkait dengan kelas, semakin besar kemungkinan instance kelas memiliki siklus hidup yang kompleks. Dengan demikian, objek-objek ini adalah kandidat untuk pemodelan status dengan mesin status perilaku.

Contoh Perumahan Kampus. Dalam Bab 4 dan 5, kami membuat satu set model fungsional dan struktural untuk layanan perumahan kampus. Pada bagian ini, kita akan menggunakan model tersebut sebagai dasar untuk melakukan analisis CRUDE. Hal pertama yang perlu kita lakukan adalah mengidentifikasi semua aktor dan kelas yang terlibat dalam contoh layanan perumahan kampus. Dalam hal ini, aktornya adalah pemilik apartemen dan siswa, dan kelasnya adalah pemilik apartemen dan apartemen. Mengingat ini, matriks CRUDE kami adalah matriks 4x4. Dalam contoh sederhana ini, kami hanya mendukung pembuatan, pembacaan, dan penghapusan instance. Secara khusus, aktor pemilik apartemen dapat membuat dan menghapus instance apartemen, sedangkan aktor pelajar dapat membaca instance pemilik apartemen dan apartemen. Gambar 6-24 menggambarkan matriks CRUDE untuk layanan perumahan kampus.

	Apartment Owner Actor	Student Actor	Apartment Owner Class	Apartment Class
Apartment Owner Actor				C,D
Student Actor			R	R
Apartment Owner Class				
Apartment Class				

Gambar 6-24 Matriks CRUDE Layanan Perumahan Kampus

	Apartment Owner Actor	Student Actor	Staff Member Actor	Apartment Owner Class	Apartment Class
Apartment Owner Actor					C,D
Student Actor				R	R
Staff Member Actor				C,D	
Apartment Owner Class					
Apartment Class					

Gambar 6-25 Matriks CRUDE Layanan Perumahan Kampus yang Dikoreksi

Namun, setelah meninjau matriks, meskipun contoh pemilik apartemen dibaca, mereka tidak pernah dibuat atau dihapus. Kecuali contoh pemilik apartemen dibuat dengan sistem lain, ini adalah situasi yang tidak mungkin. Ini adalah contoh lain mengapa kami mengikuti pendekatan iteratif dan inkremental dalam pengembangan sistem berorientasi objek. Dalam hal ini, dengan membuat matriks CRUDE, kami menemukan persyaratan tambahan yang sebelumnya diabaikan. Akibatnya, kita perlu kembali dan menambahkan Use-Case tambahan yang menambah dan menghapus pemilik apartemen yang terkait dengan aktor anggota staf layanan perumahan kampus tambahan yang menjalankannya (lihat Gambar 6-25). Jelas, pada saat ini kita harus memodifikasi diagram use-case; menambahkan diagram aktivitas; menambahkan diagram urutan; menambahkan diagram komunikasi; dan tinjau diagram kelas, kartu CRC, dan mesin status perilaku untuk memastikan bahwa semuanya masih benar. Kami akan menyerahkan modifikasi tersebut kepada Anda dan melanjutkan ke masalah perpustakaan yang telah kami gunakan dalam bab ini dan bab sebelumnya.

	Student Actor	Faculty/ Staff Actor	Guest Actor	Librarian Actor	Personnel Office Actor	Registrar's Office Actor	Book	Book Collection	Student Class	Faculty/ Staff Class	Guest Class	Interlibrary Loan System	Library	Storage
Student Actor				E			R,E	R				E		
Faculty/Staff Actor				E			R,E	R				E		
Guest Actor				E			R,E	R				E		
Librarian Actor	E	E	E		R,E	R,E	C,R,U,D,E	R,U,E	R,U	R,U	C,R,U,D,E	R,E		
Personnel Office Actor														
Registrar's Office Actor														
Book														
Book Collection														
Student Class														
Faculty/Staff Class														
Guest Class														
Interlibrary Loan System														
Library														
Storage														

Gambar 6-26 Matriks CRUDE untuk Sistem Pengelolaan Koleksi Buku Perpustakaan

Contoh Perpustakaan. Cara terbaik untuk membuat matriks CRUDE adalah secara konseptual menggabungkan urutan dan diagram komunikasi yang memodelkan semua skenario dari semua Use-Case dalam suatu sistem. Cara termudah untuk melakukannya adalah dengan membuat matriks kelas/aktor demi kelas/aktor kosong. Dalam hal Sistem Manajemen Koleksi Buku Perpustakaan, kami memiliki enam aktor (Mahasiswa, Fakultas/Staf, Tamu, Pustakawan, Bagian Personalia, dan Kepaniteraan) dan delapan kelas (Buku, Koleksi Buku, Mahasiswa, Fakultas/Staf, Tamu, Sistem Pinjaman Antar Perpustakaan, Perpustakaan, dan Penyimpanan). Setelah matriks ini ditata, skenario permainan peran akan menunjukkan aktor dan kelas mana yang berinteraksi satu sama lain. Berdasarkan jenis interaksi, catat C, R, U, D, atau E dalam sel matriks yang sesuai. Lakukan ini berulang kali hingga semua skenario dari semua Use-Case telah dieksekusi. Matriks CRUDE untuk Sistem Manajemen Koleksi Buku Perpustakaan ditunjukkan pada Gambar 6-26. Salah satu fungsi yang dapat dilayani oleh matriks adalah untuk memulai proses validasi seluruh sistem. Dalam hal ini, dengan meninjau matriks dengan cepat, kita dapat melihat bahwa sama sekali tidak ada yang berinteraksi dengan objek Library dan Storage. Hal ini menimbulkan pertanyaan penting, apakah benda-benda ini harus ada atau tidak. Jika tidak ada yang memanggil atau menggunakannya dan mereka tidak memanggil atau menggunakan apa pun, lalu mengapa mereka menjadi bagian dari sistem ini? Entah mereka harus dihapus dari representasi sistem saat ini, atau kami telah berhasil melewatkan beberapa interaksi. Mengetahui hal ini memungkinkan kita untuk kembali ke pengguna, dalam hal ini Pustakawan, dan menanyakan apa yang harus dilakukan.

6.7 VERIFIKASI DAN VALIDASI MODEL PERILAKU

Dalam bab ini, kami menggambarkan tiga diagram yang berbeda (diagram urutan, diagram komunikasi, dan mesin status perilaku) dan matriks CRUDE yang dapat digunakan untuk mewakili model perilaku. Diagram urutan dan komunikasi memodelkan interaksi antara instance kelas yang bekerja sama untuk mendukung proses bisnis yang termasuk dalam suatu sistem, mesin status perilaku menggambarkan perubahan status yang dilalui objek selama masa pakainya, dan matriks CRUDE mewakili sistem- gambaran tingkat interaksi di antara objek-objek dalam sistem. Dalam bab ini, kami menggabungkan penelusuran dengan matriks CRUDE untuk memverifikasi dan memvalidasi model perilaku secara lebih lengkap. Karena kami telah membahas analisis CRUDE dan matriks di bagian sebelumnya, kami hanya fokus

pada penelusuran di bagian ini. Kami kembali menggunakan sistem penunjukan dan fokus pada Gambar 6-1, 6-10, 6-16, dan 6-23 untuk menggambarkan seperangkat aturan yang dapat digunakan untuk memastikan bahwa model perilaku konsisten secara internal.

Pertama, setiap aktor dan objek yang termasuk dalam diagram urutan harus dimasukkan sebagai aktor dan objek pada diagram komunikasi, dan sebaliknya. Misalnya, pada Gambar 6-1 dan 6-10, aktor aReceptionist dan objek Pasien muncul di kedua diagram.

Kedua, jika ada pesan pada sequence diagram, maka harus ada asosiasi pada komunikasi diagram, dan sebaliknya. Misalnya, Gambar 6-1 menggambarkan pesan yang dikirim dari aktor aReceptionist ke objek Pasien, dan asosiasi yang cocok muncul dalam diagram komunikasi yang sesuai (lihat Gambar 6-10).

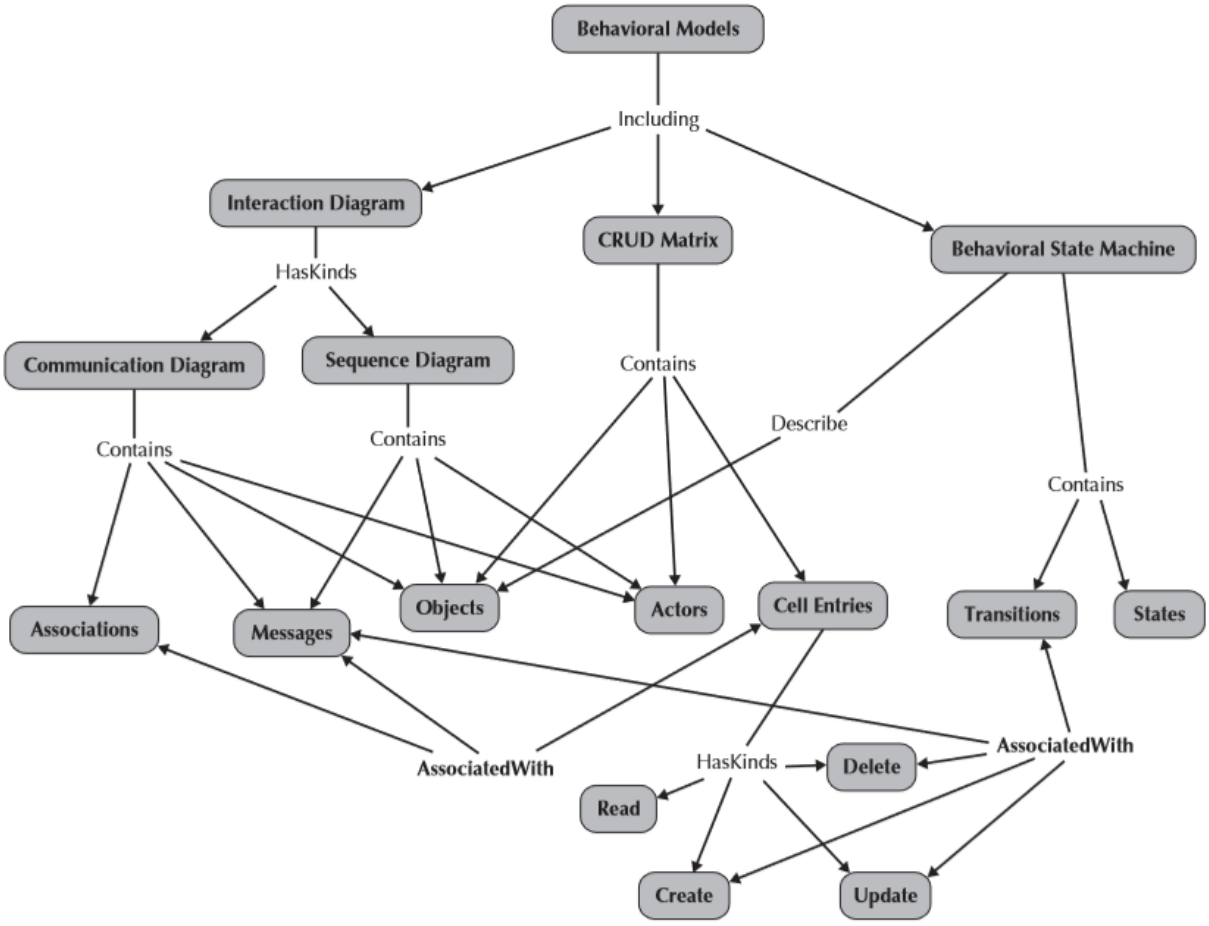
Ketiga, setiap pesan yang disertakan pada diagram urutan harus muncul sebagai pesan pada asosiasi dalam diagram komunikasi yang sesuai, dan sebaliknya. Misalnya, pesan LookUpPatient() yang dikirim oleh aktor aReceptionist ke objek Patient pada diagram urutan (lihat Gambar 6-1) muncul sebagai pesan tentang hubungan antara aktor aReceptionist dan objek Patient pada diagram komunikasi (lihat Gambar 6-10).

Keempat, jika kondisi penjaga muncul pada pesan di diagram urutan, harus ada kondisi penjaga yang setara pada diagram komunikasi yang sesuai, dan sebaliknya. Misalnya, pesan yang dikirim dari aktor aReceptionist ke objek UnpaidBills memiliki kondisi penjaga [aPatient Exists] (lihat Gambar 6-1). Gambar 6-10 menunjukkan kondisi pelindung yang cocok yang disertakan pada diagram komunikasi.

Kelima, nomor urut yang dimasukkan sebagai bagian dari label pesan dalam diagram komunikasi menyiratkan urutan berurutan di mana pesan akan dikirim. Oleh karena itu, harus sesuai dengan urutan top-down dari pesan yang dikirim pada diagram urutan. Misalnya, pesan LookUpPatient yang dikirim dari aktor aReceptionist ke objek Patient pada diagram urutan (lihat Gambar 6-1) adalah yang kedua dari atas diagram. Pesan LookUpPatient yang dikirim dari aktor aReceptionist ke objek Pasien pada diagram komunikasi (lihat Gambar 6-10) diberi label dengan nomor 2.

Keenam, semua transisi yang terkandung dalam mesin status perilaku harus dikaitkan dengan pesan yang dikirim pada diagram urutan dan komunikasi, dan harus diklasifikasikan sebagai pesan (C)reate, (U)pdate, atau (D)elete di matriks MENTAH. Misalnya, pada Gambar 6-16, transisi Checks In harus dikaitkan dengan pesan dalam urutan dan diagram komunikasi yang sesuai. Selain itu, harus dikaitkan dengan entri (U)pdate dalam matriks CRUDE yang terkait dengan sistem pasien rumah sakit.

Ketujuh, semua entri dalam matriks CRUDE menyiratkan pesan yang dikirim dari aktor atau objek ke aktor atau objek lain. Jika entri adalah (C)reate, (U)pdate, atau (D)elete, maka harus ada transisi terkait dalam mesin status perilaku yang mewakili instance dari kelas penerima. Misalnya pada Gambar 6-23 entri R dan U di baris Resepsionis dan kolom Janji temu menyiratkan bahwa instance dari aktor Resepsionis akan membaca dan memperbarui instance kelas Janji Temu. Dengan demikian, harus ada pesan yang dibaca dan diperbarui pada urutan dan diagram komunikasi yang sesuai dengan proses janji temu. Meninjau Gambar 6-1 dan 6-10, kita melihat bahwa ada pesan, MatchAppts(), dari aktor aReceptionist ke objek Appointments. Namun, berdasarkan tinjauan ini, tidak jelas apakah pesan MatchAppts() mewakili pembacaan, pembaruan, atau keduanya. Oleh karena itu, diperlukan analisis tambahan. Karena ada pesan (U)pdate yang terlibat, harus ada transisi pada mesin status perilaku yang menggambarkan siklus hidup objek Janji Temu.



Gambar 6-27 Hubungan Antar Model Perilaku

Akhirnya, banyak aturan khusus representasi telah diusulkan. Namun, seperti pada model lainnya, aturan ini berada di luar cakupan bagian verifikasi dan validasi ini. Gambar 6-27 menggambarkan asosiasi di antara model perilaku.

Pertanyaan

1. Bagaimana pemodelan perilaku terkait dengan pemodelan fungsional dan struktural?
2. Bagaimana use case berhubungan dengan sequence diagram? Diagram komunikasi?
3. Bandingkan rangkaian istilah berikut: keadaan, perilaku, kelas, objek, tindakan, dan aktivitas.
4. Mengapa iterasi penting saat membuat model perilaku?
5. Apa blok bangunan utama untuk diagram urutan? Bagaimana mereka diwakili pada model?
6. Bagaimana Anda menunjukkan bahwa objek sementara akan menghilang pada diagram urutan?
7. Apakah garis hidup selalu berlanjut ke seluruh halaman diagram urutan? Menjelaskan.
8. Jelaskan langkah-langkah yang digunakan untuk membuat diagram urutan.
9. Saat menggambar diagram urutan, pedoman apa yang harus Anda ikuti?
10. Jelaskan blok bangunan utama untuk diagram komunikasi dan bagaimana mereka diwakili pada model.
11. Bagaimana Anda menunjukkan urutan pesan pada diagram komunikasi?
12. Bagaimana Anda menunjukkan arah pesan pada diagram komunikasi?
13. Jelaskan langkah-langkah yang digunakan untuk membuat diagram komunikasi.
14. Saat menggambar diagram komunikasi, pedoman apa yang harus Anda ikuti?

15. Apakah status selalu digambarkan menggunakan persegi panjang bulat pada mesin status perilaku? Menjelaskan.
16. Jenis peristiwa apa yang dapat menyebabkan transisi status pada mesin status perilaku?
17. Apa langkah-langkah dalam membangun mesin status perilaku?
18. Saat menggambar mesin status perilaku, pedoman apa yang harus Anda ikuti?
19. Bagaimana kondisi penjaga ditampilkan pada mesin status perilaku?
20. Jelaskan jenis kelas yang paling baik diwakili oleh mesin status perilaku. Berikan dua contoh kelas yang akan menjadi kandidat yang baik untuk mesin status perilaku.
21. Apa itu analisis CRUDE, dan untuk apa digunakan?
22. Identifikasi model yang berisi setiap komponen berikut: aktor, asosiasi, kelas, perluasan, asosiasi, status akhir, kondisi penjaga, status awal, tautan, pesan, multiplisitas, objek, status, transisi, dan operasi pembaruan.

Latihan

A. Pikirkan tentang mengirim surat kelas satu ke sahabat pena internasional. Jelaskan proses yang dilalui surat itu dari pembuatan awal surat Anda hingga dibaca oleh teman Anda, dari sudut pandang surat itu. Gambarlah mesin status perilaku yang menggambarkan status yang dilalui surat tersebut.
B. Gambarkan mesin status perilaku yang menjelaskan berbagai status yang dapat dimiliki oleh otorisasi perjalanan melalui proses persetujuannya. Formulir otorisasi perjalanan digunakan di sebagian besar perusahaan untuk menyetujui biaya perjalanan bagi karyawan. Biasanya, seorang karyawan mengisi formulir kosong dan mengirimkannya ke atasannya untuk ditandatangani. Jika jumlahnya cukup kecil (<Rp. 5.400.000), maka bos menandatangani formulir dan mengarahkannya ke hutang untuk dimasukkan ke dalam sistem akuntansi. Sistem memotong cek yang dikirim ke karyawan dengan jumlah yang tepat, dan setelah cek diuangkan, formulir tersebut dibawa pergi dengan cek yang dibatalkan. Jika cek tidak diuangkan dalam waktu 90 hari, formulir perjalanan kedaluwarsa. Ketika jumlah voucher perjalanan banyak (>Rp. 5.400.000), maka bos menandatangani formulir dan mengirimkannya ke CFO, bersama dengan paragraf yang menjelaskan tujuan perjalanan; CFO menandatangani formulir dan menyerahkannya ke hutang usaha. Tentu saja, bos dan CFO dapat menolak formulir otorisasi perjalanan jika mereka merasa biayanya tidak masuk akal. Dalam hal ini, karyawan dapat mengubah formulir untuk memasukkan lebih banyak penjelasan atau memutuskan untuk membayar biaya.
C. Pikirkan tentang sistem yang menangani penerimaan mahasiswa di universitas Anda. Fungsi utama sistem harus dapat melacak siswa dari permintaan informasi melalui proses penerimaan sampai siswa diterima atau ditolak.
    1. Tulis deskripsi use-case yang dapat menggambarkan use case Admit Student.

       Asumsikan pelamar yang merupakan anak alumni ditangani berbeda dengan pelamar lainnya. Juga, asumsikan bahwa Use-Case Perbarui Informasi Siswa generik tersedia untuk digunakan sistem Anda.

    2. Buat diagram Use-Case yang mencakup semua Use-Case di atas.

       Asumsikan bahwa formulir penerimaan mencakup isi formulir, informasi SAT, dan referensi. Informasi tambahan diambil tentang anak-anak alumni, seperti tahun kelulusan orang tua mereka, informasi kontak, dan jurusan kuliah.

    3. Buat diagram kelas untuk Use-Case yang diidentifikasi dengan pertanyaan 1 dan 2. Juga, pastikan untuk menyertakan informasi di atas.

Asumsikan bahwa objek siswa sementara digunakan oleh sistem untuk menyimpan informasi tentang orang-orang sebelum mereka mengirimkan formulir penerimaan. Setelah formulir dikirim, orang-orang ini dianggap sebagai siswa.

4. Buat diagram urutan untuk skenario Use-Case di atas.
5. Buat diagram komunikasi untuk skenario Use-Case di atas.
6. Buat mesin status perilaku untuk menggambarkan seseorang saat dia bergerak melalui proses penerimaan.
7. Lakukan analisis CRUDE untuk menunjukkan interaktivitas objek dalam sistem.

D. Untuk masalah A Real Estate Inc. di Bab 4 (latihan I, J, dan K) dan 5 (latihan P dan Q):
   1. Pilih satu Use-Case dan, untuk setiap skenario, buat diagram urutan.
   2. Buat diagram komunikasi untuk setiap skenario use case yang dipilih pada Pertanyaan 1.
   3. Buat mesin status perilaku untuk menggambarkan salah satu kelas pada diagram kelas yang Anda buat untuk Bab 5, latihan P.
   4. Lakukan analisis CRUDE untuk menunjukkan interaktivitas objek dalam sistem.
   5. Lakukan langkah-langkah verifikasi dan validasi masalah.

E. Untuk masalah A Video Store di Bab 4 (latihan L, M, dan N) dan 5 (latihan R dan S):
   1. Pilih satu use case dan, untuk setiap skenario, buat diagram urutan
   2. Buat diagram komunikasi untuk setiap skenario use case yang dipilih pada Pertanyaan 1.
   3. Buat mesin status perilaku untuk menggambarkan salah satu kelas pada diagram kelas yang Anda buat untuk Bab 5, latihan R.
   4. Lakukan analisis CRUDE untuk menunjukkan interaktivitas objek dalam sistem.
   5. Lakukan penelusuran verifikasi dan validasi masalah.

F. Untuk masalah keanggotaan gym di Bab 4 (latihan O, P, dan Q) dan 5 (latihan T dan U):
   1. Pilih satu Use-Case dan, untuk setiap skenario, buat diagram urutan.
   2. Buat diagram komunikasi untuk setiap skenario use case yang dipilih dalam Pertanyaan 1.
   3. Buat mesin status perilaku untuk menggambarkan salah satu kelas pada diagram kelas yang Anda buat untuk Bab 5, latihan T.
   4. Lakukan analisis CRUDE untuk menunjukkan interaktivitas objek dalam sistem.
   5. Lakukan penelusuran verifikasi dan validasi masalah.

G. Untuk masalah Piknik R Us di Bab 4 (latihan R, S, dan T) dan 5 (latihan V dan W):
   1. Pilih satu Use-Case dan, untuk setiap skenario, buat diagram urutan.
   2. Buat diagram komunikasi untuk setiap skenario dari use case yang dipilih pada Pertanyaan 1.
   3. Buat mesin status perilaku untuk menggambarkan salah satu kelas pada diagram kelas yang Anda buat untuk Bab 5, latihan V.
   4. Lakukan analisis CRUDE untuk menunjukkan interaktivitas objek dalam sistem.
   5. Lakukan penelusuran verifikasi dan validasi masalah.

H. Untuk masalah Klub Bulanan di Bab 4 (latihan U, V, dan W) dan 5 (latihan X dan Y):
   1. Pilih satu Use-Case dan, untuk setiap skenario, buat diagram urutan.

2. Buat diagram komunikasi untuk setiap skenario use case yang dipilih pada Pertanyaan 1.
3. Buat mesin status perilaku untuk menggambarkan salah satu kelas pada diagram kelas yang Anda buat untuk Bab 5, latihan X.
4. Lakukan analisis CRUDE untuk menunjukkan interaktivitas objek dalam sistem.
5. Lakukan langkah-langkah verifikasi dan validasi masalah.

Minicase

1. Lihat model fungsional (diagram Use-Case, diagram aktivitas, dan deskripsi Use-Case) yang Anda siapkan untuk Minicase Professional and Scientific Staff Management (PSSM) di Bab 4. Berdasarkan kinerja Anda, PSSM sangat puas sehingga ingin Anda untuk mengembangkan model struktural dan perilaku sehingga dapat lebih memahami interaksi yang akan terjadi antara pengguna dan sistem dan sistem itu sendiri secara lebih rinci.
    a. Buat kartu CRC dan diagram kelas berdasarkan model fungsional yang dibuat di Bab 4.
    b. Buat urutan dan diagram komunikasi untuk setiap skenario dari setiap Use-Case yang diidentifikasi dalam model fungsional.
    c. Buat mesin status perilaku untuk setiap kelas kompleks dalam diagram kelas.
    d. Lakukan analisis CRUDE untuk menunjukkan interaktivitas objek dalam sistem.
    e. Lakukan penelusuran verifikasi dan validasi setiap model: fungsional, struktural, dan perilaku.
2. Lihat model struktural (kartu CRC dan diagram kelas) yang Anda buat untuk Minicase Kendaraan Perjalanan Liburan di Bab 5. Berdasarkan kinerja Anda, Kendaraan Perjalanan Liburan sangat puas sehingga ingin Anda mengembangkan model fungsional dan perilaku sehingga itu bisa lebih memahami baik interaksi yang akan terjadi antara pengguna dan sistem dan sistem itu sendiri secara lebih rinci.
    a. Berdasarkan model struktural yang Anda buat di Bab 5 dan deskripsi masalah di Bab 5, buat model fungsional (diagram use case, diagram aktivitas, dan deskripsi use case) untuk proses bisnis yang terkait dengan sistem penjualan Kendaraan Perjalanan Liburan.
    b. Buat urutan dan diagram komunikasi untuk setiap skenario dari setiap Use-Case yang diidentifikasi dalam model fungsional.
    c. Buat mesin status perilaku untuk setiap kelas kompleks dalam diagram kelas.
    d. Lakukan analisis CRUDE untuk menunjukkan interaktivitas objek dalam sistem.
    e. Lakukan walkthrough verifikasi dan validasi masing-masing model: fungsional, struktural, dan behavioral.

BAGIAN 2

PEMODELAN DESAIN

Pemodelan analisis berkonsentrasi pada persyaratan fungsional dari sistem yang berkembang, pemodelan desain menggabungkan persyaratan nonfungsional. Artinya, pemodelan desain berfokus pada bagaimana sistem akan beroperasi. Pertama, tim proyek memverifikasi dan memvalidasi model analisis (fungsional, struktural, dan perilaku). Selanjutnya, satu set model analisis terfaktor dan dipartisi dibuat. Desain kelas dan metode diilustrasikan menggunakan spesifikasi kelas (menggunakan kartu CRC dan diagram kelas), kontrak, dan spesifikasi metode. Selanjutnya, lapisan manajemen data ditangani dengan merancang database aktual atau struktur file yang akan digunakan untuk persistensi objek, dan satu set kelas yang akan memetakan spesifikasi kelas ke dalam format persistensi objek yang dipilih. Secara bersamaan, tim menghasilkan desain lapisan interface (antarmuka) pengguna menggunakan skenario penggunaan, diagram navigasi windows, Use-Case nyata, template antarmuka, storyboard, diagram tata letak windows, dan prototipe antarmuka pengguna. Desain lapisan arsitektur fisik dibuat menggunakan diagram penyebaran dan spesifikasi perangkat lunak perangkat keras. Kumpulan kiriman merupakan spesifikasi sistem yang diserahkan kepada tim pemrograman untuk implementasi.

BAB 7

PINDAH KE DESAIN

Pengembangan sistem berorientasi objek menggunakan persyaratan yang dikumpulkan selama analisis guna membuat cetak biru untuk sistem masa depan. Desain berorientasi objek yang sukses dibangun di atas apa yang dipelajari pada fase sebelumnya dan mengarah pada implementasi yang mulus dengan membuat rencana yang jelas dan akurat tentang apa yang perlu dilakukan. Bab ini menjelaskan transisi awal dari analisis ke desain dan menyajikan tiga cara untuk mendekati desain untuk sistem baru.

7.1 Tujuan

- Memahami verifikasi dan validasi model analisis.
- Memahami transisi dari analisis ke desain.
- Memahami penggunaan factoring, partisi, dan lapisan.
- Mampu membuat diagram paket. Kenali alternatif desain kustom, paket, dan outsourcing. Mampu membuat matriks alternatif.

7.2 Pendahuluan

Tujuan analisis adalah untuk mengetahui apa kebutuhan bisnis. Tujuan dari desain adalah untuk memutuskan bagaimana membangun sistem. Aktivitas utama yang terjadi selama desain adalah mengembangkan himpunan representasi analisis menjadi representasi desain.

Melalui desain kasar, tim proyek dengan hati-hati mempertimbangkan sistem baru sehubungan dengan lingkungan saat ini dan sistem yang ada di dalam organisasi secara keseluruhan. Pertimbangan utama dalam menentukan bagaimana sistem akan bekerja mencakup faktor lingkungan, seperti mengintegrasikan dengan sistem yang ada, mengubah data dari sistem lama, dan memanfaatkan keterampilan yang ada di rumah. Meskipun

perencanaan dan analisis dilakukan untuk mengembangkan sistem yang mungkin, tujuan desain adalah membuat cetak biru untuk sistem yang dapat diimplementasikan.

Bagian awal yang penting dari desain adalah memeriksa beberapa strategi desain dan memutuskan mana yang akan digunakan untuk membangun sistem. Sistem dapat dibangun dari awal, dibeli dan disesuaikan, atau dialihdayakan ke pihak lain, dan tim proyek perlu menyelidiki kelayakan setiap alternatif. Keputusan ini mempengaruhi tugas-tugas yang harus diselesaikan selama desain.

Pada saat yang sama, desain rinci dari masing-masing kelas dan metode yang digunakan untuk memetakan mur dan baut sistem dan cara penyimpanannya masih harus diselesaikan. Teknik seperti kartu CRC, diagram kelas, spesifikasi kontrak, spesifikasi metode, dan desain database memberikan detail desain akhir dalam persiapan untuk fase implementasi, dan teknik tersebut memastikan bahwa pemrogram memiliki informasi yang cukup untuk membangun sistem yang tepat secara efisien.

Desain juga mencakup aktivitas seperti mendesain antarmuka pengguna, input sistem, dan output sistem, yang melibatkan cara pengguna berinteraksi dengan sistem. Bab 10 menjelaskan ketiga aktivitas ini secara rinci, bersama dengan teknik seperti storyboard dan prototyping, yang membantu tim proyek merancang sistem yang memenuhi kebutuhan penggunanya dan memuaskan untuk digunakan.

Akhirnya, keputusan arsitektur fisik dibuat mengenai perangkat keras dan perangkat lunak yang akan dibeli untuk mendukung sistem baru dan cara pemrosesan sistem akan diatur. Misalnya, sistem dapat diatur sehingga pemrosesannya terpusat di satu lokasi, terdistribusi, atau keduanya terpusat dan terdistribusi, dan setiap solusi menawarkan manfaat dan tantangan unik bagi tim proyek. Karena isu global dan keamanan mempengaruhi rencana implementasi yang dibuat, maka perlu dipertimbangkan bersama dengan sistem teknis arsitektur

Banyak langkah desain yang sangat saling terkait dan, seperti langkah-langkah dalam analisis, analis sering bolak-balik di antara mereka. Misalnya, prototyping dalam langkah desain antarmuka sering kali mengungkap informasi tambahan yang dibutuhkan dalam sistem. Atau, sistem yang sedang dirancang untuk organisasi yang memiliki sistem terpusat mungkin memerlukan investasi perangkat keras dan perangkat lunak yang substansial jika tim proyek memutuskan untuk mengubah ke sistem di mana semua pemrosesan didistribusikan.

Tips Praktis

Menghindari Desain Klasik

Dalam Bab 2, kita membahas beberapa kesalahan klasik dan bagaimana cara menghindarinya. Di sini, kami merangkum empat kesalahan klasik dalam desain dan mendiskusikan cara menghindarinya.

1. *Mengurangi waktu desain:* Jika waktunya singkat, ada godaan untuk mengurangi waktu yang dihabiskan dalam kegiatan "tidak produktif" seperti desain sehingga tim dapat terjun ke pemrograman "produktif". Hal ini mengakibatkan hilangnya rincian penting yang harus diselidiki kemudian pada waktu dan biaya yang jauh lebih tinggi (biasanya setidaknya sepuluh kali lebih tinggi).

   Solusi: Jika tekanan waktu sangat tinggi, gunakan timeboxing untuk menghilangkan fungsionalitas atau memindahkannya ke versi mendatang.

2. *Fitur creep*: Bahkan jika Anda berhasil menghindari scope creep, sekitar 25 persen persyaratan sistem masih akan berubah. Dan, perubahan—besar dan kecil—dapat secara signifikan meningkatkan waktu dan biaya.

   Solusi: Pastikan bahwa semua perubahan sangat penting dan pengguna menyadari dampaknya terhadap biaya dan waktu. Cobalah untuk memindahkan perubahan yang diusulkan ke versi yang akan datang.

3. *Silver Bullet Syndrom*: Analis terkadang percaya klaim pemasaran untuk beberapa alat desain yang mengklaim dapat menyelesaikan semua masalah dan secara ajaib mengurangi waktu dan biaya. Tidak ada satu alat atau teknik yang dapat menghilangkan keseluruhan waktu atau biaya lebih dari 25 persen (walaupun beberapa dapat mengurangi langkah individu sebanyak ini).

   Solusi: Jika alat desain memiliki klaim yang tampaknya terlalu bagus untuk menjadi kenyataan, katakan saja tidak.

4. *Mengganti alat di tengah proyek*: Terkadang analis beralih ke alat yang tampaknya lebih baik selama desain dengan harapan menghemat waktu atau biaya. Biasanya, manfaat apa pun sebanding dengan kebutuhan untuk mempelajari alat baru. Ini juga berlaku bahkan untuk peningkatan kecil pada alat saat ini.

   Solusi: Jangan beralih atau memutakhirkan kecuali ada kebutuhan mendesak untuk fitur khusus di alat baru, dan kemudian secara eksplisit menambah jadwal untuk menyertakan waktu belajar.

Sumber : Berdasarkan materi dari Steve McConnell, Rapid Development (Redmond, WA: Microsoft Press, 1966).

Verifikasi dan Validasi Model Analisis

Sebelum kita mengembangkan representasi analisis kita menjadi representasi desain, kita perlu memverifikasi dan memvalidasi rangkaian model analisis saat ini untuk memastikan bahwa model tersebut dengan tepat mewakili domain masalah yang sedang dipertimbangkan. Ini termasuk menguji kesetiaan setiap model; misalnya, kita harus yakin bahwa diagram aktivitas, deskripsi Use-Case, dan diagram Use-Case semuanya menggambarkan persyaratan fungsional yang sama. Ini juga melibatkan pengujian kesetiaan antara model; misalnya, transisi pada mesin status perilaku dikaitkan dengan operasi yang terkandung dalam diagram kelas. Dalam Bab 4, 5, dan 6, kami berfokus pada verifikasi dan validasi model individu: fungsi, struktural, dan perilaku. Dalam bab ini, kami memusatkan perhatian kami untuk memastikan bahwa model yang berbeda konsisten. Gambar 7-1 menggambarkan fakta bahwa model analisis berorientasi objek sangat saling terkait. Misalnya, apakah model fungsional dan struktural setuju? Bagaimana dengan model fungsional dan perilaku? Dan akhirnya, apakah model struktural dan perilaku dapat dipercaya? Di bagian ini, kami menjelaskan seperangkat aturan yang berguna untuk memverifikasi dan memvalidasi perpotongan model analisis. Tergantung pada konstruksi spesifik dari setiap model aktual, keterkaitan yang berbeda relevan. Proses memastikan konsistensi di antara mereka dikenal sebagai penyeimbangan model.

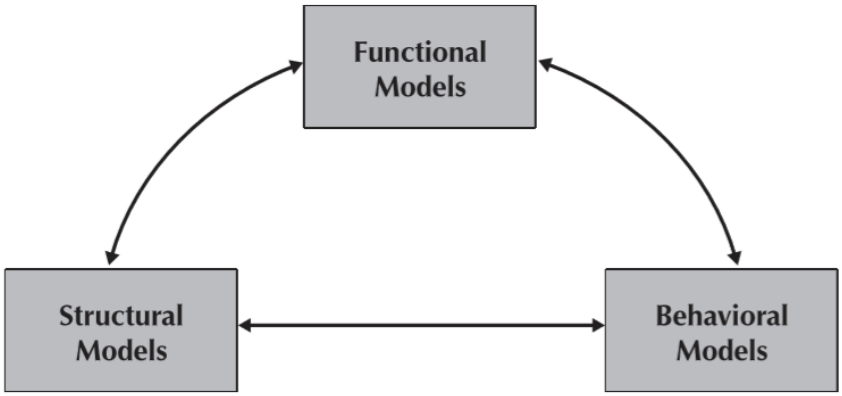
Menyeimbangkan Model Fungsional dan Struktural

Untuk menyeimbangkan model fungsional dan struktural, kita harus memastikan bahwa kedua set model konsisten satu sama lain. Artinya, diagram aktivitas, deskripsi Use-Case, dan diagram Use-Case harus sesuai dengan kartu CRC dan diagram kelas yang mewakili model yang berkembang dari domain masalah. Gambar 7-2 menunjukkan keterkaitan antara

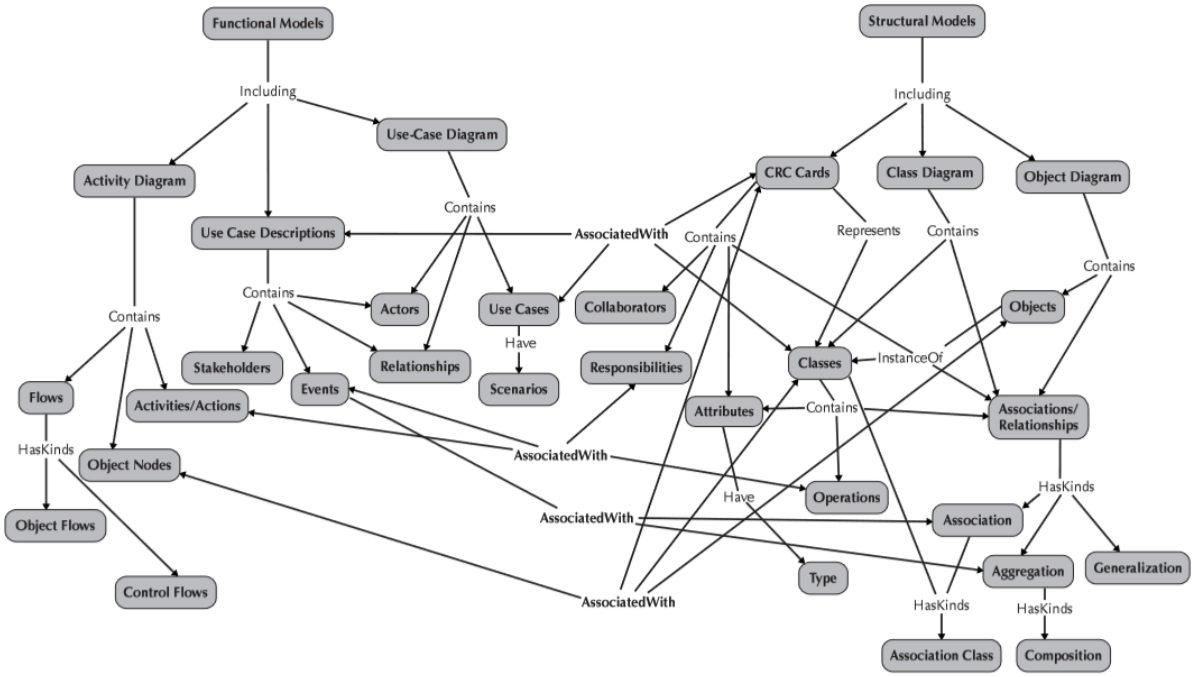
model fungsional dan struktural. Dengan meninjau gambar ini, kita dapat menemukan empat set asosiasi antara model. Ini memberi kita tempat untuk mulai menyeimbangkan model fungsional dan struktural.

Pertama, setiap kelas pada diagram kelas dan setiap kartu CRC harus dikaitkan dengan setidaknya satu use case, dan sebaliknya. Misalnya, kartu CRC yang digambarkan pada Gambar 7-3 dan kelas terkait yang terdapat dalam diagram kelas (lihat Gambar 7-4) dikaitkan dengan Use-Case Aplikasi Make Old Patient yang dijelaskan pada Gambar 7-5.

Kedua, setiap aktivitas atau tindakan yang terdapat dalam diagram aktivitas (lihat Gambar 7-6) dan setiap peristiwa yang terdapat dalam deskripsi Use-Case (lihat Gambar 7-5) harus terkait dengan satu atau lebih tanggung jawab pada kartu CRC dan satu atau lebih banyak operasi di kelas pada diagram kelas dan sebaliknya. Misalnya, aktivitas Dapatkan Informasi Pasien pada diagram aktivitas contoh (lihat Gambar 7-6) dan dua peristiwa pertama pada deskripsi Use-Case (lihat Gambar 7-5) dikaitkan dengan tanggung jawab membuat janji temu pada kartu CRC (lihat Gambar 7-3) dan operasi makeAppointment() di kelas Pasien pada diagram kelas (lihat Gambar 7-4).



Gambar 7-1 Model Analisis Berorientasi Objek



Gambar 7-2 Hubungan antara Model Fungsional dan Struktural

Ketiga, setiap node objek pada diagram aktivitas harus dikaitkan dengan instance kelas pada diagram kelas (yaitu, objek) dan kartu CRC atau atribut yang terdapat dalam kelas dan kartu CRC. Namun, pada Gambar 7-6, ada node objek, Appt Request Info, yang tampaknya tidak terkait dengan kelas mana pun dalam diagram kelas yang digambarkan pada Gambar 7-4. Dengan demikian, aktivitas atau diagram kelas salah atau simpul objek harus mewakili atribut. Dalam hal ini, tampaknya tidak mewakili atribut. Kita bisa menambahkan kelas ke diagram kelas yang membuat objek sementara yang terkait dengan simpul objek pada diagram aktivitas. Namun, tidak jelas operasi apa, jika ada, yang akan dikaitkan dengan objek sementara ini. Oleh karena itu, solusi yang lebih baik adalah menghapus node objek Appt Request Info dari diagram aktivitas. Pada kenyataannya, node objek ini hanya mewakili sekumpulan nilai atribut yang dibundel, yaitu data yang akan digunakan dalam proses sistem penunjukan (lihat Gambar 7-7).

Keempat, setiap atribut dan hubungan asosiasi/agregasi yang terdapat pada kartu CRC (dan terhubung ke kelas pada diagram kelas) harus terkait dengan subjek atau objek dari suatu peristiwa dalam deskripsi Use-Case. Misalnya, pada Gambar 7-5, kejadian kedua menyatakan: Pasien memberikan nama dan alamatnya kepada Resepsionis. Dengan meninjau kartu CRC pada Gambar 7-3 dan diagram kelas pada Gambar 7-4, kita melihat bahwa kelas Pasien adalah subkelas dari kelas Peserta dan karenanya mewarisi semua atribut, asosiasi, dan operasi yang didefinisikan dengan kelas Peserta , di mana atribut nama dan alamat didefinisikan.

Menyeimbangkan Model Fungsional dan Perilaku

Seperti dalam menyeimbangkan model fungsional dan struktural, kita harus memastikan konsistensi dari dua set model. Dalam hal ini, diagram aktivitas, deskripsi Use-Case, dan diagram Use-Case harus sesuai dengan diagram urutan, diagram komunikasi, mesin status perilaku, dan matriks CRUDE. Gambar 7-8 menggambarkan hubungan antara model fungsional dan model perilaku. Berdasarkan hubungan timbal balik ini, kita melihat bahwa ada empat bidang yang harus kita perhatikan.[33]

[33] Melakukan analisis CRUDE (lihat Bab 6) juga dapat berguna dalam meninjau persimpangan antara model fungsional dan perilaku behavioral

Front:

Class Name: Patient	**ID:** 3	**Type:** Concrete, Domain
Description: An individual who needs to receive or has received medical attention		**Associated Use Cases:** 2

Responsibilities	Collaborators
Make appointment	Appointment
Calculate last visit	
Change status	
Provide medical history	Medical history

Back:

Attributes:

Amount (double)	
Insurance carrier (text)	

Relationships:

Generalization (a-kind-of): Participant

Aggregation (has-parts):

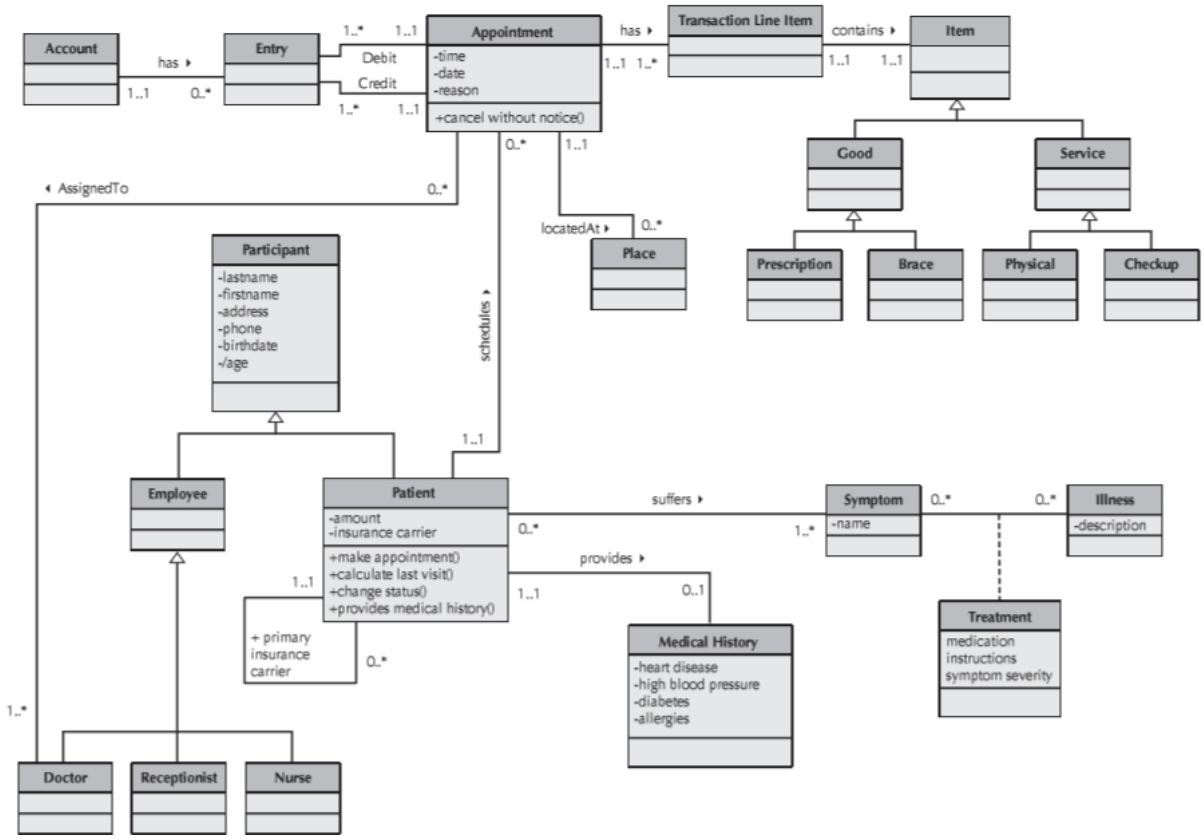
Other Associations: Appointment, Medical History

Gambar 7-3 Kartu CRC Pasien Lama (Gambar 5-25)

Pertama, diagram urutan dan komunikasi harus dikaitkan dengan use case pada diagram use case dan deskripsi use case. Sebagai contoh, sequence diagram pada Gambar 7-9 dan komunikasi diagram pada Gambar 7-10 terkait dengan skenario use case Make Old Patient Appt yang muncul pada deskripsi use case pada Gambar 7-5 dan use case diagram pada Gambar 7-11.

Kedua, aktor pada diagram urutan, diagram komunikasi, dan/atau matriks CRUDE harus dikaitkan dengan aktor pada diagram use-case atau direferensikan dalam deskripsi usecase, dan sebaliknya. Misalnya, aktor aPatient dalam diagram urutan pada Gambar 7-9, diagram komunikasi pada Gambar 7-10, dan baris dan kolom Pasien dalam matriks CRUDE pada Gambar 7-12 muncul dalam diagram use-case pada Gambar 7-11 dan deskripsi kasus pada Gambar 7-5. Namun, aReceptionist tidak muncul dalam diagram Use-Case tetapi

direferensikan dalam peristiwa yang terkait dengan deskripsi Use-Case Aplikasi Pasien Lama. Dalam hal ini, aktor aReceptionist jelas merupakan aktor internal, yang tidak dapat digambarkan pada diagram use-case UML.



Gambar 7-4 Diagram Kelas Masalah Janji Temu (Gambar 5-7)

Ketiga, pesan pada diagram urutan dan komunikasi, transisi pada mesin status perilaku, dan entri dalam matriks CRUDE harus terkait dengan aktivitas dan tindakan pada diagram aktivitas dan peristiwa yang tercantum dalam deskripsi Use-Case, dan sebaliknya. Misalnya, pesan CreateAppt() pada diagram urutan dan komunikasi (lihat Gambar 7-9 dan 7-10) terkait dengan aktivitas CreateAppointment (lihat Gambar 7-7) dan subflow S-1: New Appointment pada penggunaan -deskripsi kasus (lihat Gambar 7-5). Entri C dalam sel Pengangkatan Resepsionis dari matriks CRUDE juga dikaitkan dengan pesan, aktivitas, dan subalur ini.

Keempat, semua objek kompleks yang diwakili oleh node objek dalam diagram aktivitas harus memiliki mesin status perilaku yang mewakili siklus hidup objek, dan sebaliknya. Sebagaimana dinyatakan dalam Bab 6, objek kompleks cenderung sangat dinamis dan melewati berbagai keadaan selama masa hidupnya. Namun, dalam kasus ini karena kita tidak lagi memiliki node objek dalam diagram aktivitas (lihat Gambar 7-7), tidak ada keharusan untuk membuat mesin status perilaku berdasarkan diagram aktivitas.

Gunakan Nama Kasus	: Buat Aplikasi Pasien Lama	ID: 2	Tigkat Kepentingan : Rendah
Aktor Utama	: Pasien Lama	Jenis Kasus Penggunaan	: Detail, Penting

Pemangku Kepentingan:
Pasien Lama – ingin membuat, mengubah, atau membatalkan janji
Dokter – ingin memastikan kebutuhan pasien terpenuhi tepat waktu

Deskripsi singkat : Kasus penggunaan ini menjelaskan bagaimana kami membuat janji temu serta mengubah atau membatalkan janji temu untuk pasien yang ditemui sebelumnya.

Pemicu : Pasien menelepon dan meminta janji baru atau meminta untuk membatalkan atau mengubah janji yang sudah ada

Jenis : Eksternal

Hubungan :
- Asosiasi : Pasien Lama
- Termasuk :
- Perluas : Perbarui Informasi Pasien
- Generalisasi : Kelola Janji Temu

Alur Peristiwa Normal:
1. Pasien menghubungi kantor mengenai janji temu.
2. Pasien memberikan nama dan alamatnya kepada Resepsionis.
3. Jika informasi Pasien telah berubah
   Jalankan kasus penggunaan Perbarui Informasi Pasien.
4. Jika pengaturan pembayaran Pasien telah berubah
   Jalankan use case Lakukan Pengaturan Pembayaran.
5. Resepsionis bertanya kepada Pasien apakah dia ingin membuat janji baru, membatalkan janji yang sudah ada, atau mengubah janji yang sudah ada.
   Jika pasien ingin membuat janji baru,
   S-1: subflow janji baru dilakukan.
   Jika pasien ingin membatalkan janji yang sudah ada,
   S-2: pembtalan subflow janji temu dilakukan.
   Jika pasien ingin mengubah janji temu yang ada,
   S-3: perubahan subflow janji dilakukan.
6. Resepsionis memberikan hasil transaksi kepada Pasien.

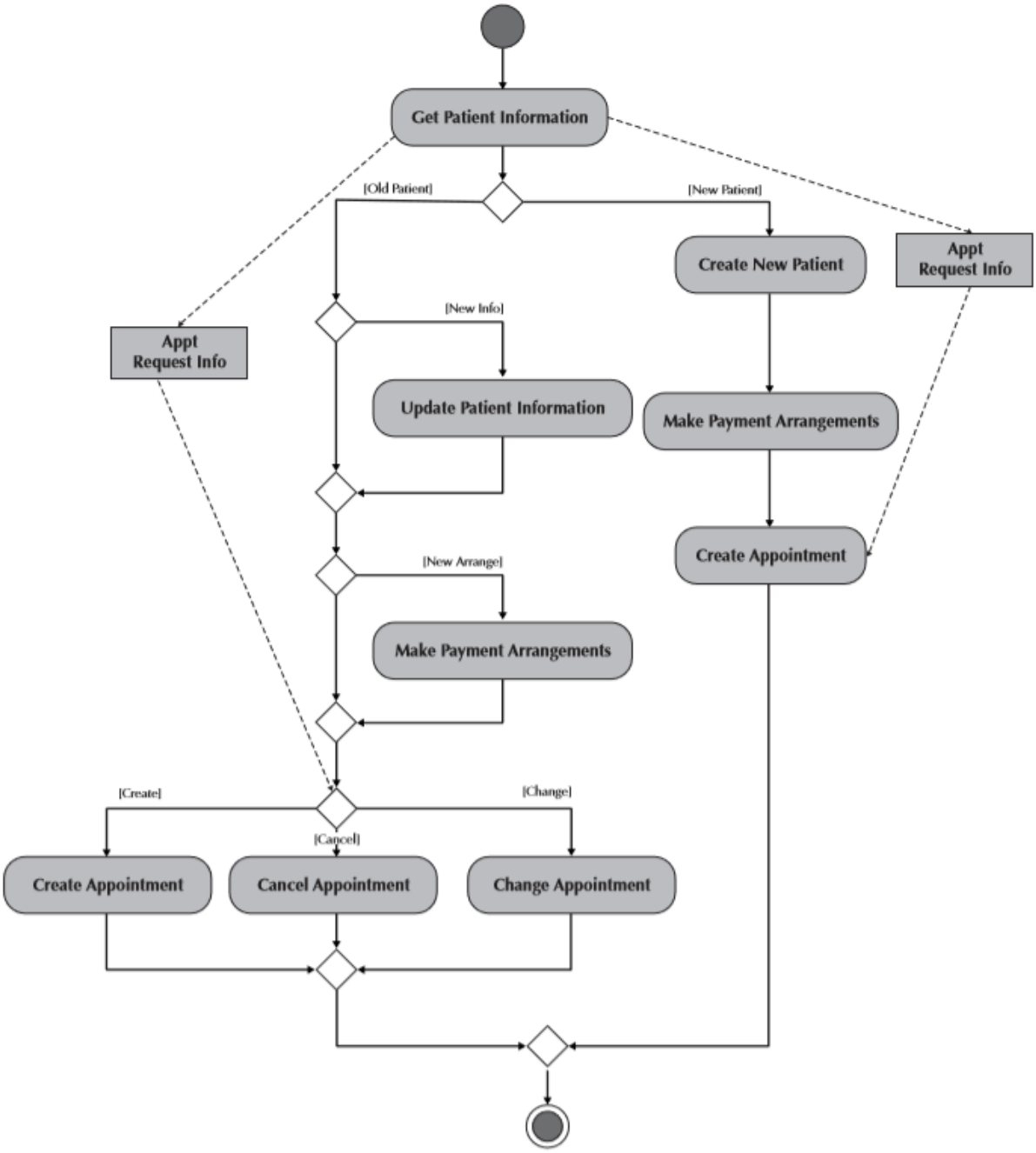
Sub Aliran:
S-1: Janji Baru
1. Resepsionis menanyakan kemungkinan waktu janji temu kepada Pasien.
2. Resepsionis mencocokkan waktu janji temu yang diinginkan Pasien dengan tanggal dan waktu yang tersedia serta menjadwalkan janji temu baru.

S-2: Batalkan Janji Temu
1. Resepsionis menanyakan waktu janji lama kepada Pasien.
2. Resepsionis menemukan janji temu saat ini di file janji temu dan membatalkannya.

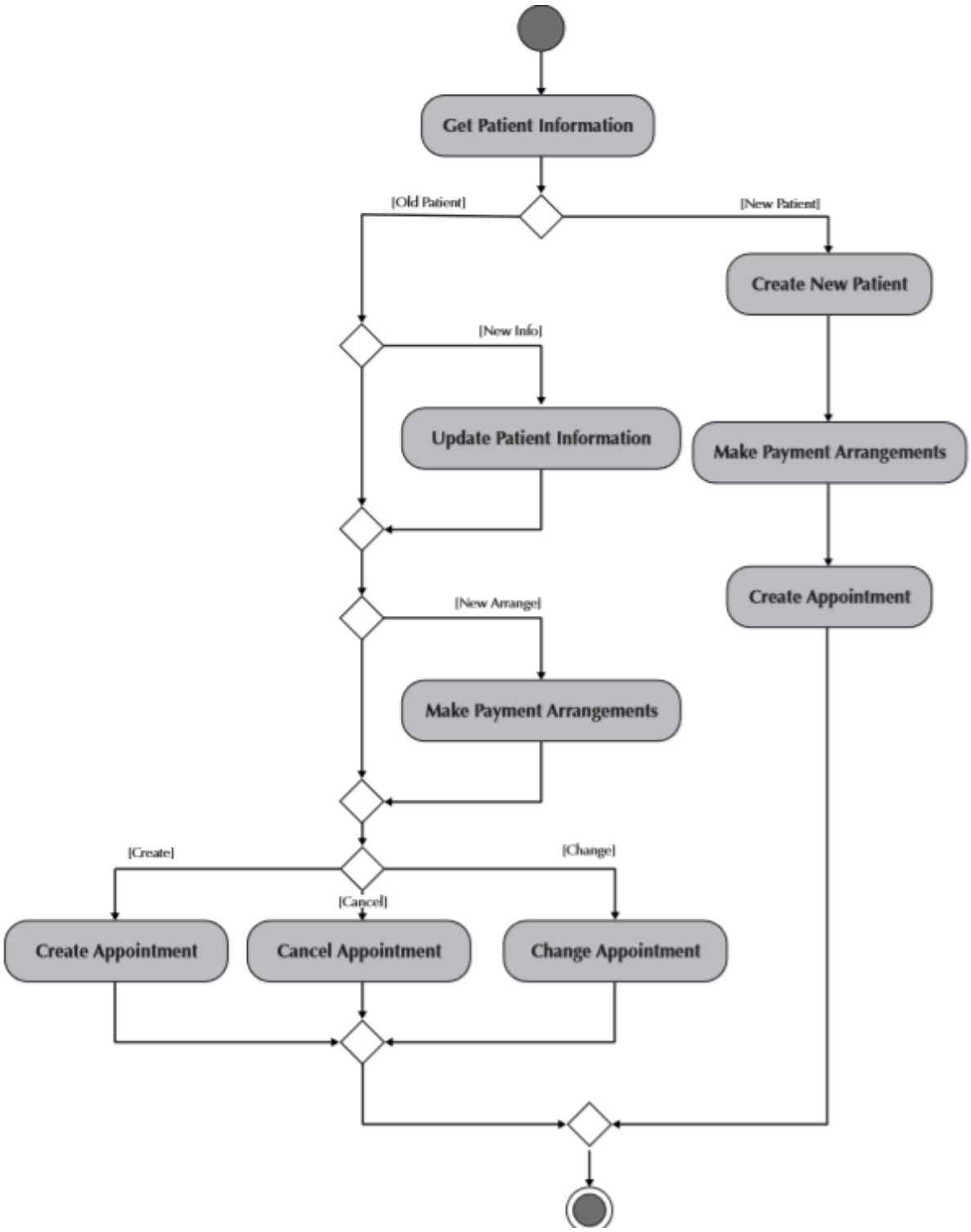
S-3: Ubah Janji Temu
1. Resepsionis melakukan subflow S-2: batalkan janji temu.
2. Resepsionis melakukan S-1: subflow janji baru.

Aliran Alternatif/Aliran Pengecualian:
S-1, 2a1: Resepsionis mengusulkan beberapa alternatif waktu janji temu berdasarkan yang tersedia di jadwal janji temu.
S-1, 2a2: Pasien memilih salah satu waktu yang diusulkan atau memutuskan untuk tidak membuat janji.

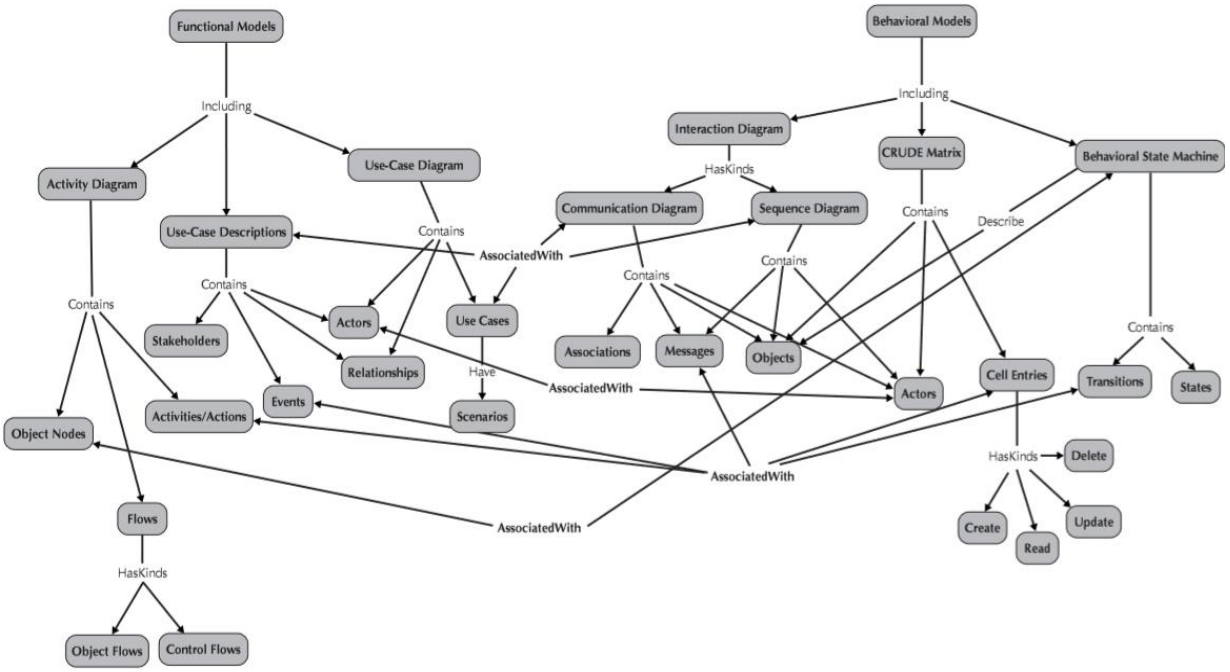
Gambar 7-5 Deskripsi Use-Case untuk Use-Case Aplikasi Make Old Patient (Gambar 4-13)



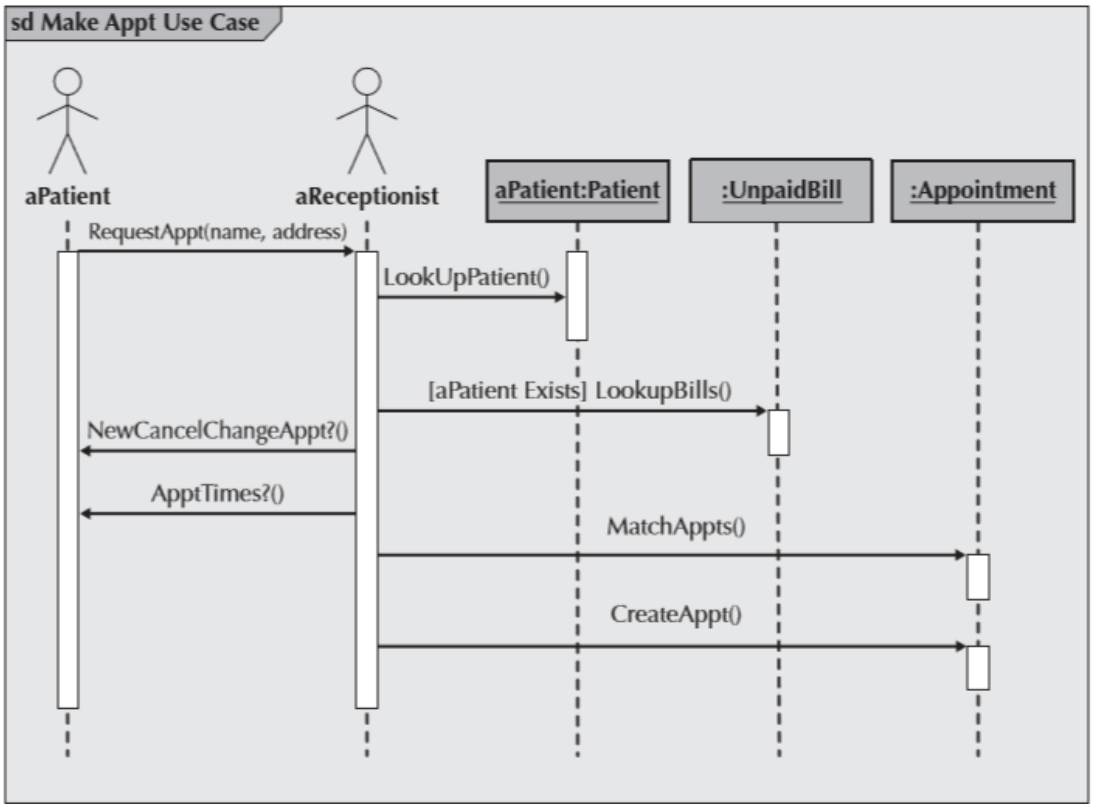
Gambar 7-6 Diagram Aktivitas untuk Use-Case Kelola Janji Temu (Gambar 4-8)



Gambar 7-7 Diagram Aktivitas yang Dikoreksi untuk Use-Case Kelola Janji Temu



Gambar 7-8 Hubungan antara Model Fungsional dan Perilaku

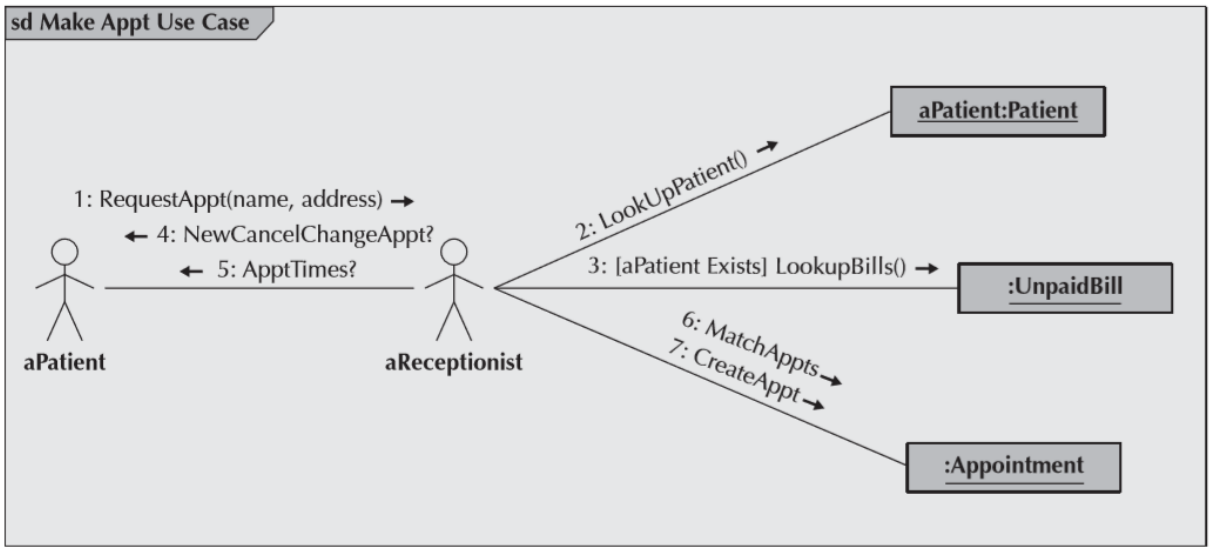


Gambar 7-9 Diagram Urutan untuk Skenario Use-Case Aplikasi Make Old Patient

Menyeimbangkan Model Struktural dan Perilaku

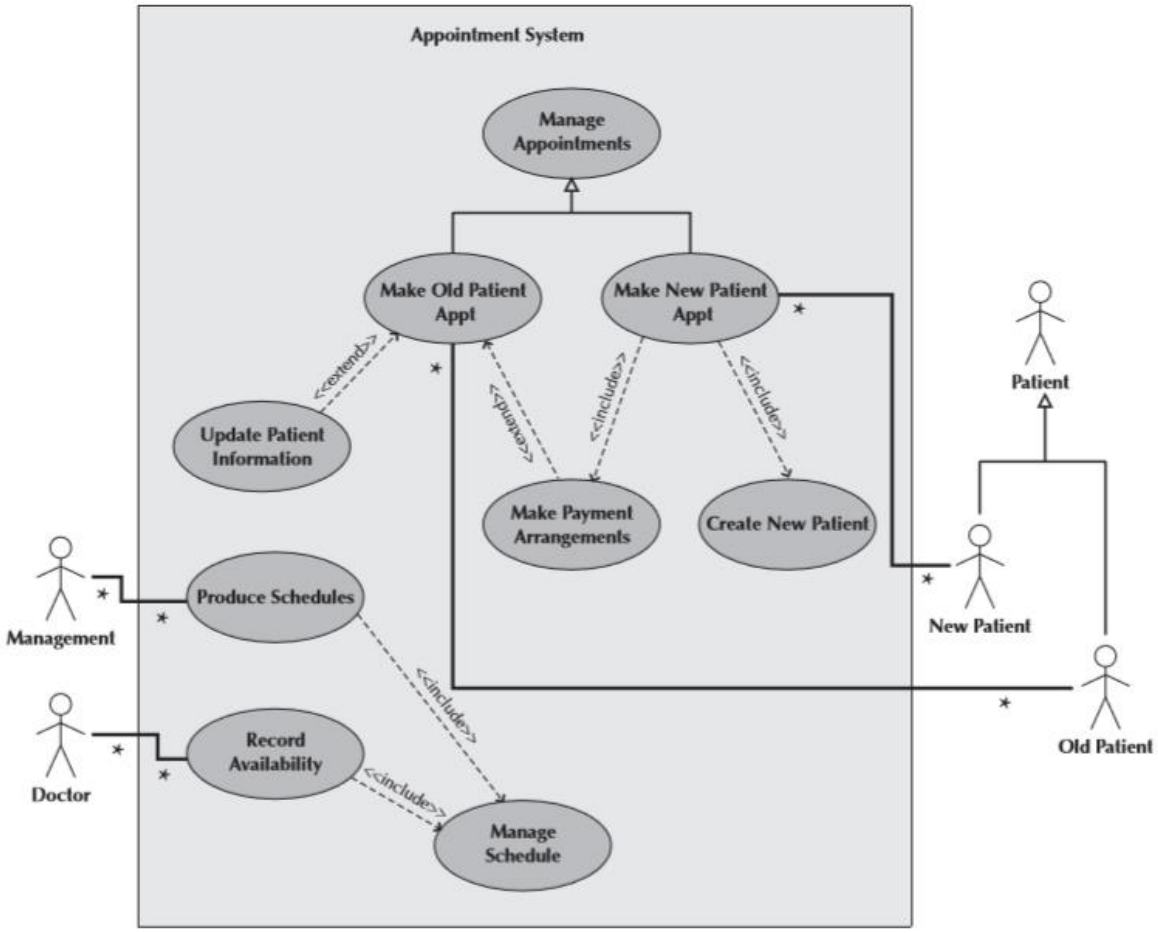
Untuk menemukan hubungan antara model struktural dan model perilaku, kami menggunakan peta konsep pada Gambar 7-13. Dalam hal ini, ada lima area di mana kita harus memastikan konsistensi antar model.[34]

Pertama, objek yang muncul dalam matriks CRUDE harus dikaitkan dengan kelas yang diwakili oleh kartu CRC dan muncul pada diagram kelas, dan sebaliknya. Misalnya, kelas Pasien dalam matriks CRUDE pada Gambar 7-12 dikaitkan dengan kartu CRC pada Gambar 7-3 dan kelas Pasien dalam diagram kelas pada Gambar 7-4.



Gambar 7-10 Diagram Komunikasi untuk Skenario Use-Case Aplikasi Make Old Patient (Gambar 6-10)

[34] Bermain peran (lihat Bab 5) dan analisis CRUDE (lihat Bab 6) juga bisa sangat berguna dalam usaha ini.



Gambar 7-11 Diagram Use-Case yang Dimodifikasi untuk Sistem Penunjukan (Gambar 4-21)

Kedua, karena mesin status perilaku mewakili siklus hidup objek kompleks, mereka harus dikaitkan dengan instance (objek) kelas pada diagram kelas dan dengan kartu CRC yang mewakili kelas instance. Misalnya, mesin status perilaku yang menjelaskan turunan dari kelas Pasien pada Gambar 7-14 menyiratkan bahwa kelas Pasien ada pada diagram kelas terkait (lihat Gambar 7-4) dan bahwa kartu CRC ada untuk kelas terkait (lihat Gambar 7-3).

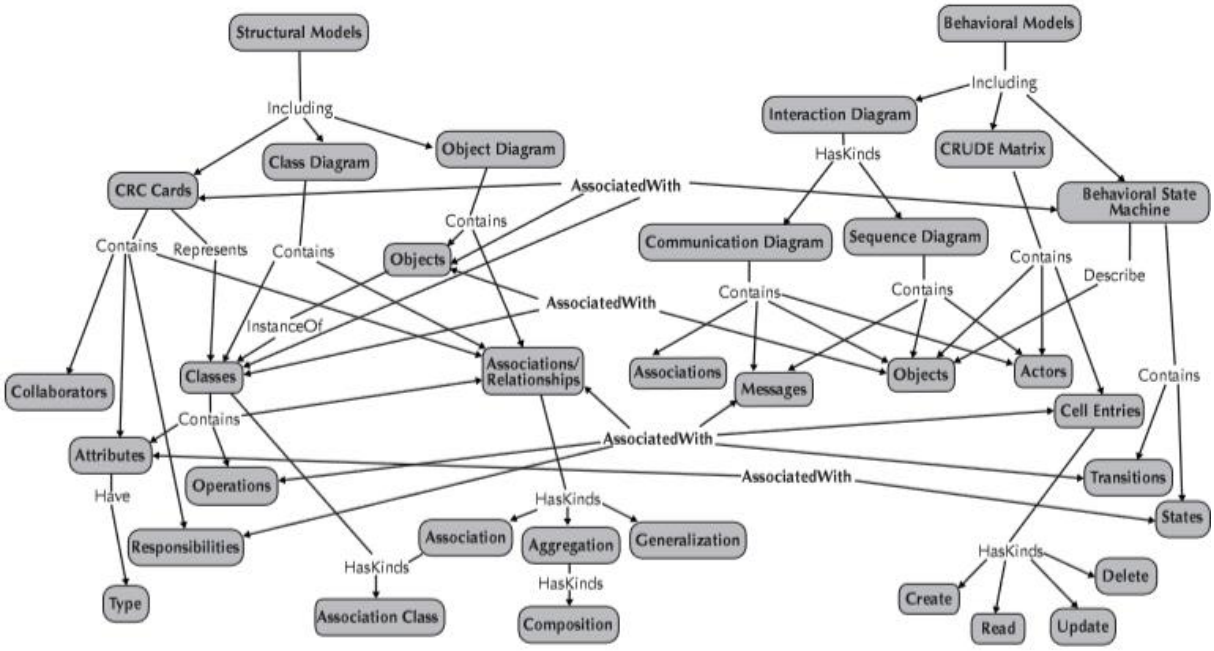
Ketiga, komunikasi dan diagram urutan berisi objek yang harus menjadi instantiasi kelas yang diwakili oleh kartu CRC dan terletak pada diagram kelas. Misalnya, Gambar 7-9 dan Gambar 7-10 memiliki objek anAppt yang merupakan instantiasi dari kelas Appointment. Oleh karena itu, kelas Appointment harus ada dalam diagram kelas (lihat Gambar 7-4), dan harus ada kartu CRC yang menjelaskannya. Namun, ada objek pada diagram komunikasi dan urutan yang terkait dengan kelas yang tidak ada pada diagram kelas: UnpaidBill. Pada titik ini, analis harus memutuskan untuk memodifikasi diagram kelas dengan menambahkan kelas-kelas ini atau memikirkan kembali komunikasi dan diagram urutan. Dalam hal ini, lebih baik menambahkan kelas ke diagram kelas (lihat Gambar 7-15).

Keempat, pesan yang terkandung pada diagram urutan dan komunikasi, transisi pada mesin status perilaku, dan entri sel pada matriks CRUDE harus dikaitkan dengan tanggung jawab dan asosiasi pada kartu CRC dan operasi di kelas dan asosiasi yang terhubung ke kelas pada diagram kelas. Misalnya, pesan CreateAppt() pada diagram urutan dan komunikasi (lihat Gambar 7-9 dan 7-10) berhubungan dengan operasi makeAppointment dari kelas Patient dan asosiasi jadwal antara kelas Patient dan Appointment pada diagram kelas (lihat Gambar 7-15).

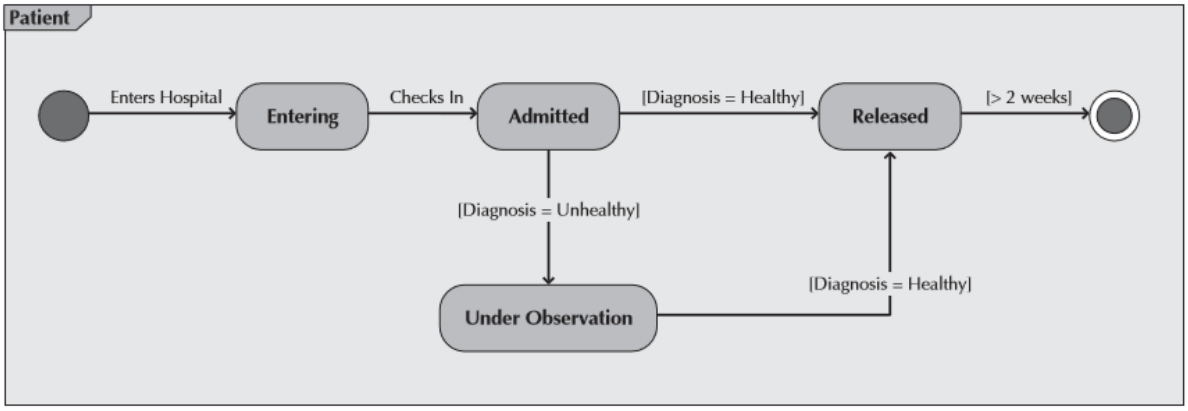
Kelima , keadaan dalam mesin keadaan perilaku harus dikaitkan dengan nilai yang berbeda dari atribut atau set atribut yang menggambarkan suatu objek. Misalnya, mesin status perilaku untuk objek pasien rumah sakit menyiratkan bahwa harus ada atribut, kemungkinan status saat ini, yang perlu dimasukkan dalam definisi kelas.

	Receptionist	PatientList	Patient	UnpaidBills	Appointments	Appointment
Receptionist		RU	CRUD	R	RU	CRUD
PatientList			R			
Patient						
UnpaidBills						
Appointments						R
Appointment						

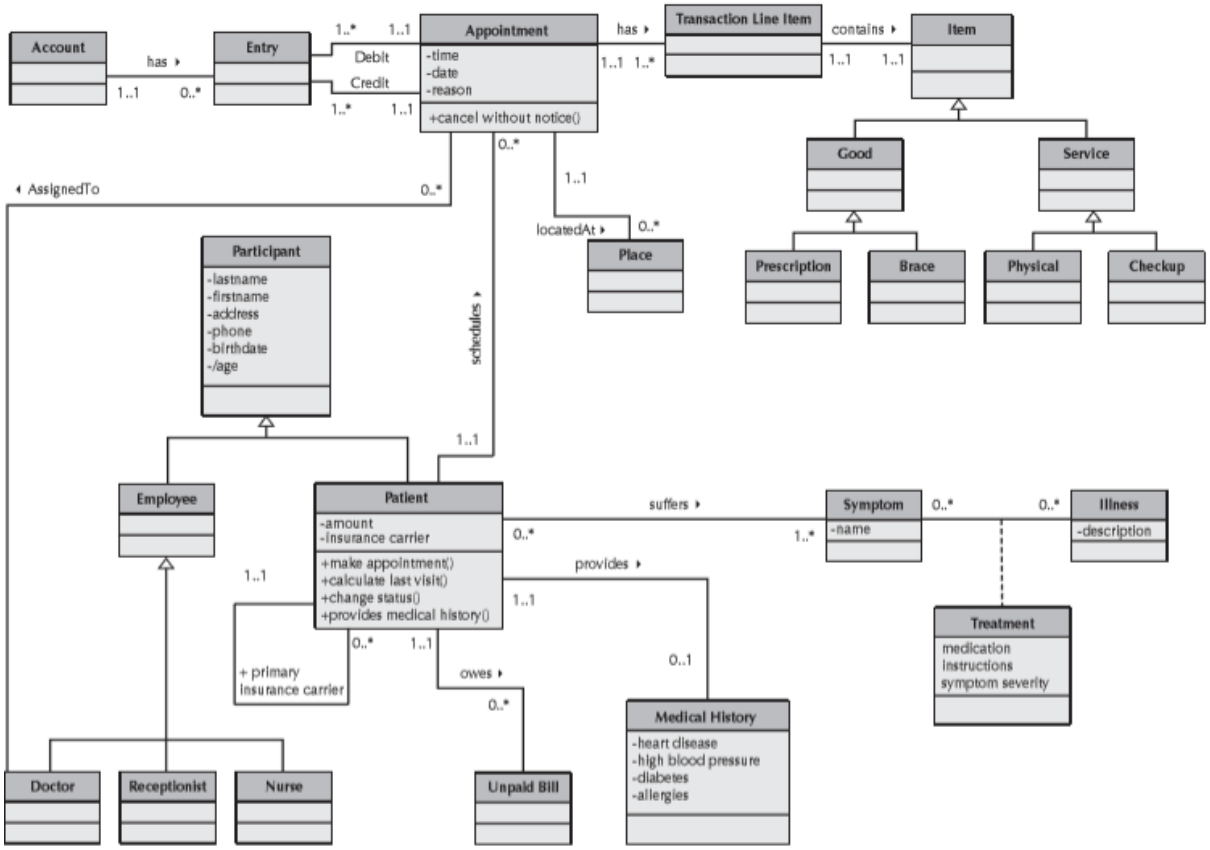
Gambar 7-12 Matriks CRUDE untuk Use-Case Make Old Patient Apt (Gambar 6-23)



Gambar 7-13 Hubungan antara Model Struktural dan Perilaku



Gambar 7-14 Mesin Status Perilaku untuk Pasien Rumah Sakit (Gambar 6-16)



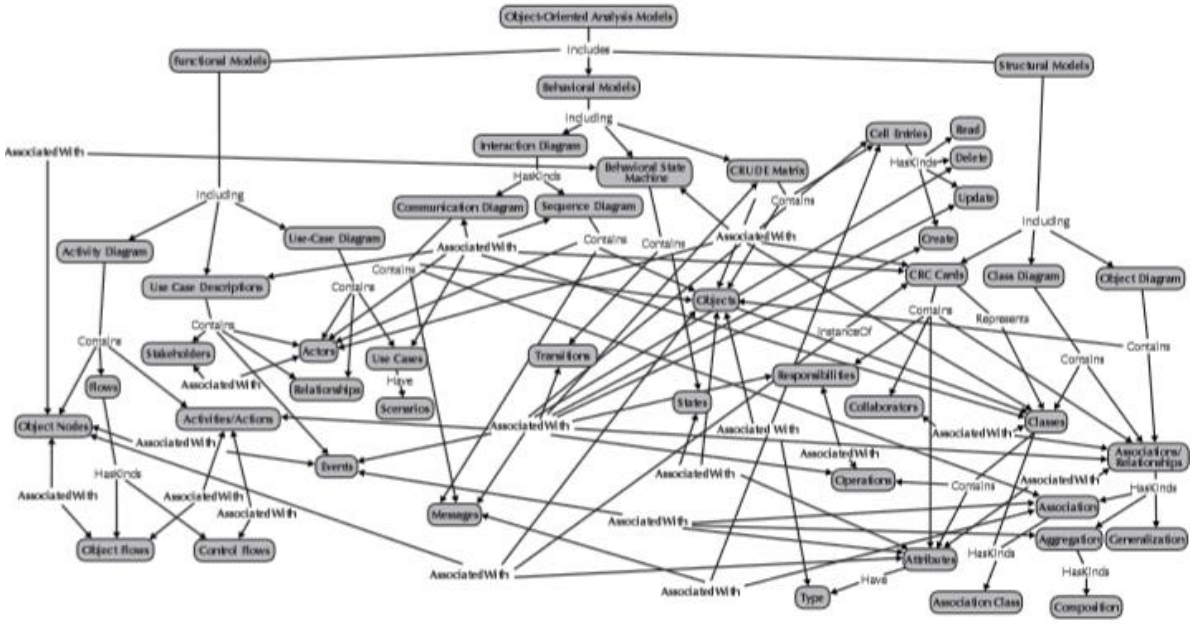
Gambar 7.-15 Diagram Kelas Sistem Pengangkatan yang Dikoreksi

Ringkasan

Gambar 7-16 menggambarkan peta konsep yang merupakan gambaran lengkap dari keterkaitan antar diagram yang dibahas dalam bagian ini. Jelas dari kompleksitas gambar ini bahwa menyeimbangkan semua model fungsional, struktural, dan perilaku adalah tugas yang sangat memakan waktu, membosankan, dan sulit. Namun, tanpa memberikan tingkat perhatian pada model yang berkembang yang mewakili sistem, model tidak akan memberikan dasar yang kuat untuk merancang dan membangun sistem.

7.3 MENGEMBANGKAN MODEL ANALISIS MENJADI MODEL DESAIN

Sekarang setelah kami berhasil memverifikasi dan memvalidasi model analisis kami, kami perlu mulai mengembangkannya menjadi model desain yang sesuai. Tujuan dari model analisis adalah untuk mewakili domain masalah bisnis yang mendasari sebagai satu set objek yang berkolaborasi. Dengan kata lain, kegiatan analisis mendefinisikan kebutuhan fungsional. Untuk mencapai hal ini, kegiatan analisis mengabaikan persyaratan nonfungsional seperti kinerja dan masalah lingkungan sistem (misalnya, pemrosesan terdistribusi atau terpusat, masalah antarmuka pengguna, dan masalah database). Sebaliknya, tujuan utama dari model desain adalah untuk meningkatkan kemungkinan berhasil memberikan sistem yang mengimplementasikan persyaratan fungsional dengan cara yang terjangkau dan mudah dipelihara. Oleh karena itu, dalam desain sistem, kami membahas persyaratan fungsional dan nonfungsional.



Gambar 7-16 Keterkaitan antara Model Analisis Berorientasi Objek

Dari perspektif berorientasi objek, model desain sistem hanya menyempurnakan model analisis sistem dengan menambahkan detail lingkungan sistem (atau domain solusi) ke dalamnya dan menyempurnakan informasi domain masalah yang sudah terkandung dalam model analisis. Saat mengembangkan model analisis ke dalam model desain, Anda harus terlebih dahulu meninjau Use-Case dan kumpulan kelas saat ini dengan cermat (operasi dan atributnya serta hubungan di antara mereka). Apakah semua kelas diperlukan? Apakah ada kelas yang hilang? Apakah kelas sepenuhnya ditentukan? Apakah ada atribut atau metode yang hilang? Apakah kelas memiliki atribut dan metode yang tidak perlu? Apakah representasi saat ini dari sistem yang berkembang optimal? Jelas, jika kita telah memverifikasi dan memvalidasi model analisis, cukup banyak yang telah terjadi. Namun, pengembangan sistem berorientasi objek bersifat inkremental dan iteratif. Oleh karena itu, kita harus meninjau kembali model analisis. Namun, kali ini kita mulai melihat model domain masalah melalui lensa desain. Pada langkah ini, kami melakukan modifikasi pada model domain masalah yang akan meningkatkan efisiensi dan efektivitas sistem yang berkembang.

Pada bagian berikut, kami memperkenalkan pemfaktoran, partisi dan kolaborasi, dan lapisan sebagai cara untuk mengembangkan model analisis berorientasi domain masalah menjadi model desain berorientasi domain solusi optimal. Dari perspektif Proses Terpadu yang disempurnakan (lihat Gambar 1-16), kami bergerak dari alur kerja analisis ke alur kerja desain, dan kami bergerak lebih jauh ke fase Elaborasi dan sebagian ke fase Konstruksi.

Factoring

Factoring adalah proses memisahkan modul menjadi modul yang berdiri sendiri. Modul baru dapat berupa kelas baru atau metode baru. Misalnya, ketika meninjau satu set kelas, mungkin ditemukan bahwa mereka memiliki satu set atribut dan metode yang serupa. Dengan demikian, mungkin masuk akal untuk memfaktorkan kesamaan ke dalam kelas yang terpisah. Bergantung pada apakah kelas baru harus berada dalam hubungan superclass ke kelas yang ada atau tidak, kelas baru dapat dikaitkan dengan kelas yang ada melalui generalisasi (a-kind-of) atau mungkin melalui hubungan agregasi (memiliki-bagian). Menggunakan contoh sistem janji temu, jika kelas Karyawan belum diidentifikasi, kita mungkin dapat mengidentifikasinya pada tahap ini dengan memfaktorkan metode dan atribut yang serupa dari kelas Perawat, Resepsionis, dan Dokter. Dalam hal ini, kita akan menghubungkan kelas baru

(Karyawan) ke kelas yang ada menggunakan hubungan generalisasi (a-kind-of). Jelas, dengan ekstensi kami juga bisa membuat kelas Partisipan jika belum diidentifikasi sebelumnya.

Abstraksi dan penyempurnaan adalah dua proses yang terkait erat dengan anjak piutang. Abstraksi berkaitan dengan penciptaan ide tingkat yang lebih tinggi dari serangkaian ide. Mengidentifikasi kelas Karyawan adalah contoh abstraksi dari satu set kelas yang lebih rendah ke yang lebih tinggi. Dalam beberapa kasus, proses abstraksi mengidentifikasi kelas abstrak, sedangkan dalam situasi lain, proses abstraksi mengidentifikasi kelas konkret tambahan. Proses penyempurnaan adalah kebalikan dari proses abstraksi. Dalam contoh sistem janji temu, kita dapat mengidentifikasi subkelas tambahan dari kelas Karyawan, seperti Sekretaris dan Pembukuan. Tentu saja kami akan menambahkan kelas baru hanya jika ada perbedaan yang cukup di antara mereka. Jika tidak, kelas yang lebih umum, Karyawan, sudah cukup.

Partisi dan Kolaborasi

Berdasarkan semua factoring, refining, dan abstracting yang dapat terjadi pada sistem yang berkembang, ukuran representasi sistem dapat membebani pengguna dan pengembang. Pada titik ini dalam evolusi sistem, mungkin masuk akal untuk membagi representasi menjadi satu set partisi. Partisi adalah ekuivalen berorientasi objek dari subsistem, di mana subsistem adalah dekomposisi dari sistem yang lebih besar ke dalam sistem komponennya (misalnya, sistem informasi akuntansi dapat didekomposisi secara fungsional menjadi sistem hutang, sistem piutang , sistem penggajian, dll.). Dari perspektif berorientasi objek, partisi didasarkan pada pola aktivitas (pesan yang dikirim) di antara objek dalam sistem berorientasi objek. Kami menjelaskan pendekatan mudah untuk memodelkan partisi dan kolaborasi nanti dalam bab ini: paket dan diagram paket.

Tempat yang baik untuk mencari partisi potensial adalah kolaborasi yang dimodelkan dalam diagram komunikasi UML. Jika Anda ingat, salah satu cara yang berguna untuk mengidentifikasi kolaborasi adalah dengan membuat diagram komunikasi untuk setiap use case. Namun, karena kelas individu dapat mendukung beberapa Use-Case, kelas individu dapat berpartisipasi dalam beberapa kolaborasi berbasis Use-Case. Dalam kasus di mana kelas mendukung beberapa Use-Case, kolaborasi harus digabungkan. Diagram kelas harus ditinjau untuk melihat bagaimana kelas yang berbeda terkait satu sama lain. Misalnya, jika atribut kelas memiliki tipe objek yang kompleks, seperti Orang, Alamat, atau Departemen, dan tipe objek ini tidak dimodelkan sebagai asosiasi dalam diagram kelas, kita perlu mengenali asosiasi tersirat ini. Membuat diagram yang menggabungkan diagram kelas dengan diagram komunikasi dapat sangat berguna untuk menunjukkan sejauh mana kelas digabungkan. Semakin besar coupling antar kelas, semakin besar kemungkinan kelas harus dikelompokkan bersama dalam kolaborasi atau partisi. Dengan melihat matriks CRUDE, kita dapat menggunakan analisis CRUDE (lihat Bab 6) untuk mengidentifikasi kelas potensial untuk menggabungkan kolaborasi.

Salah satu teknik yang paling mudah untuk mengidentifikasi kelas-kelas yang dapat dikelompokkan untuk membentuk suatu kolaborasi adalah melalui penggunaan analisis klaster atau penskalaan multidimensi. Teknik statistik ini memungkinkan tim untuk mengelompokkan kelas secara objektif berdasarkan afinitas mereka satu sama lain. Afinitas dapat didasarkan pada hubungan semantik, berbagai jenis pesan yang dikirim di antara mereka (misalnya, membuat, membaca, memperbarui, menghapus, atau mengeksekusi), atau kombinasi berbobot dari keduanya. Ada banyak ukuran kesamaan yang berbeda dan banyak algoritma berbeda yang menjadi dasar cluster, jadi seseorang harus berhati-hati saat menggunakan teknik ini. Selalu pastikan bahwa kolaborasi yang diidentifikasi menggunakan teknik ini masuk akal dari perspektif domain masalah. Hanya karena algoritme matematika

menunjukkan bahwa kelas-kelas itu milik bersama tidak membuatnya demikian. Namun, ini adalah pendekatan yang baik untuk menciptakan serangkaian kolaborasi pertama.

Tergantung pada kompleksitas kolaborasi gabungan, mungkin berguna dalam menguraikan kolaborasi menjadi beberapa partisi. Dalam hal ini, selain kolaborasi antar objek, dimungkinkan juga kolaborasi antar partisi. Aturan umumnya adalah semakin banyak pesan yang dikirim antar objek, semakin besar kemungkinan objek tersebut berada dalam partisi yang sama. Semakin sedikit pesan yang dikirim, semakin kecil kemungkinan kedua objek tersebut menjadi satu.

Pendekatan lain yang berguna untuk mengidentifikasi partisi potensial adalah dengan memodelkan setiap kolaborasi antara objek dalam hal klien, server, dan kontrak. Klien adalah turunan dari kelas yang mengirim pesan ke turunan kelas lain untuk metode yang akan dieksekusi; server adalah instance dari kelas yang menerima pesan; dan kontrak adalah spesifikasi yang memformalkan interaksi antara objek klien dan server (lihat Bab 5 dan 8). Pendekatan ini memungkinkan pengembang untuk membangun partisi potensial dengan melihat kontrak yang telah ditentukan antar objek. Dalam hal ini, semakin banyak kontrak antara objek, semakin besar kemungkinan objek berada di partisi yang sama. Semakin sedikit kontrak, semakin kecil kemungkinan kedua kelas berada di partisi yang sama.

Ingat, tujuan utama dari mengidentifikasi kolaborasi dan partisi adalah untuk menentukan kelas mana yang harus dikelompokkan bersama dalam desain.

Lapisan

Sampai saat ini dalam pengembangan sistem kami, kami hanya berfokus pada domain masalah; kami benar-benar mengabaikan lingkungan sistem (manajemen data, antarmuka pengguna, dan arsitektur fisik). Untuk berhasil mengembangkan model analisis sistem menjadi model desain sistem, kita harus menambahkan informasi lingkungan sistem. Salah satu cara yang berguna untuk melakukan ini, tanpa membebani pengembang, adalah dengan menggunakan lapisan. Lapisan mewakili elemen arsitektur perangkat lunak dari sistem yang berkembang. Kami hanya berfokus pada satu lapisan dalam arsitektur perangkat lunak yang berkembang: lapisan domain masalah. Harus ada lapisan untuk setiap elemen yang berbeda dari lingkungan sistem (misalnya, manajemen data, antarmuka pengguna, arsitektur fisik). Seperti partisi dan kolaborasi, lapisan juga dapat digambarkan menggunakan paket dan diagram paket (lihat bagian selanjutnya dari bab ini).

Ide untuk memisahkan elemen arsitektur yang berbeda ke dalam lapisan yang terpisah dapat ditelusuri kembali ke arsitektur MVC Smalltalk. Ketika Smalltalk pertama kali dibuat,[35] penulis memutuskan untuk memisahkan logika aplikasi dari logika antarmuka pengguna. Dengan cara ini, dimungkinkan untuk dengan mudah mengembangkan antarmuka pengguna yang berbeda yang bekerja dengan aplikasi yang sama. Untuk mencapai ini, mereka menciptakan arsitektur Model–View–Controller (MVC), di mana Model mengimplementasikan logika aplikasi (domain masalah) dan Views dan Controller mengimplementasikan logika untuk antarmuka pengguna. Tampilan menangani output, dan Controller menangani input. Karena antarmuka pengguna grafis pertama kali dikembangkan dalam bahasa Smalltalk, arsitektur MVC berfungsi sebagai dasar untuk hampir semua

[35] Smalltalk ditemukan pada awal 1970-an oleh tim peneliti pengembangan perangkat lunak di Xerox PARC. Ini memperkenalkan banyak ide baru ke dalam bidang bahasa pemrograman (misalnya, orientasi objek, antarmuka pengguna berbasis windows, perpustakaan kelas yang dapat digunakan kembali, dan lingkungan pengembangan). Dalam banyak hal, Smalltalk adalah induk dari semua bahasa berbasis objek dan berorientasi objek, seperti Visual Basic, C++, dan Java.

antarmuka pengguna grafis yang telah dikembangkan saat ini (termasuk antarmuka Mac, keluarga Windows, dan berbagai lingkungan GUI berbasis Unix).

Layers	Examples	Relevant Chapters
Foundation	Date, Enumeration	7, 8
Problem Domain	Employee, Customer	4, 5, 6, 7, 8
Data Management	DataInputStream, FileInputStream	8, 9
Human–Computer Interaction	Button, Panel	8, 10
Physical Architecture	ServerSocket, URLConnection	8, 11

Gambar 7-17 Lapisan dan Sample Classes

Berdasarkan arsitektur MVC inovatif Smalltalk, banyak lapisan perangkat lunak yang berbeda telah diusulkan. Kami menyarankan lapisan berikut yang menjadi dasar arsitektur perangkat lunak: fondasi, domain masalah, manajemen data, interaksi manusia-komputer, dan arsitektur fisik (lihat Gambar 7- 17). Setiap lapisan membatasi jenis kelas yang bisa ada di atasnya (misalnya, hanya kelas antarmuka pengguna yang mungkin ada di lapisan interaksi manusia-komputer).

Fondasi. ***Lapisan fondasi,*** dalam banyak hal, adalah lapisan yang sangat tidak menarik. Ini berisi kelas-kelas yang diperlukan untuk setiap aplikasi berorientasi objek ada. Mereka termasuk kelas yang mewakili tipe data dasar (misalnya, bilangan bulat, bilangan real, karakter, string), kelas yang mewakili struktur data dasar, kadang-kadang disebut sebagai kelas kontainer (misalnya, daftar, pohon, grafik, set, tumpukan, antrian), dan kelas yang mewakili abstraksi yang berguna, kadang-kadang disebut sebagai kelas utilitas (misalnya, tanggal, waktu, uang). Kelas-kelas ini jarang, jika pernah, dimodifikasi oleh pengembang. Mereka hanya digunakan. Saat ini, kelas yang ditemukan pada lapisan ini biasanya disertakan dengan lingkungan pengembangan berorientasi objek.

Domain Masalah. ***Lapisan domain-masalah*** inilah yang menjadi fokus perhatian kami hingga saat ini. Pada tahap ini dalam pengembangan sistem kami, kami perlu merinci kelas lebih lanjut sehingga kami dapat menerapkannya secara efektif dan efisien. Banyak masalah yang perlu ditangani saat mendesain kelas, tidak peduli di lapisan mana mereka muncul. Misalnya, ada masalah yang berkaitan dengan pemfaktoran, kohesi dan coupling, connascence, enkapsulasi, penggunaan pewarisan dan polimorfisme yang tepat, kendala, spesifikasi kontrak, dan desain metode terperinci. Masalah-masalah ini dibahas dalam Bab 8.

Manajemen data. ***Lapisan manajemen data*** membahas masalah yang melibatkan kegigihan objek yang terkandung dalam sistem. Jenis kelas yang muncul di lapisan ini berhubungan dengan bagaimana objek dapat disimpan dan diambil. Kelas-kelas yang terdapat dalam lapisan ini disebut kelas Akses dan Manipulasi Data (DAM). Kelas DAM memungkinkan kelas domain masalah menjadi independen dari penyimpanan yang digunakan dan, karenanya, meningkatkan portabilitas sistem yang berkembang. Beberapa masalah yang terkait dengan lapisan ini termasuk pilihan format penyimpanan dan pengoptimalan. Ada sejumlah besar pilihan yang berbeda untuk memilih untuk menyimpan objek. Ini termasuk file sekuensial, file akses acak, database relasional, database objek/relasional, database berorientasi objek, dan penyimpanan data NoSQL. Masing-masing opsi ini telah dioptimalkan untuk memberikan solusi bagi masalah akses dan penyimpanan yang berbeda. Saat ini, dari perspektif praktis, tidak ada solusi tunggal yang secara optimal melayani semua aplikasi. Solusi yang tepat kemungkinan besar adalah kombinasi dari opsi penyimpanan yang berbeda.

Penjelasan lengkap tentang semua masalah yang terkait dengan lapisan manajemen data berada di luar cakupan buku ini. Namun, kami menyajikan dasar-dasarnya di Bab 9.

Interaksi Manusia-Komputer. ***Lapisan interaksi manusia-komputer*** berisi kelas-kelas yang terkait dengan ide View dan Controller dari Smalltalk. Tujuan utama dari lapisan ini adalah untuk menjaga implementasi antarmuka pengguna tertentu terpisah dari kelas domain masalah. Ini meningkatkan portabilitas sistem yang berkembang. Kelas khas yang ditemukan pada lapisan ini termasuk kelas yang dapat digunakan untuk mewakili tombol, jendela, bidang teks, bilah gulir, kotak centang, daftar drop-down, dan banyak kelas lain yang mewakili elemen antarmuka pengguna.

Saat merancang antarmuka pengguna untuk suatu aplikasi, banyak masalah yang harus ditangani: Seberapa penting konsistensi di seluruh antarmuka pengguna yang berbeda? Bagaimana dengan tingkat pengalaman pengguna yang berbeda? Bagaimana pengguna diharapkan dapat menavigasi melalui sistem? Bagaimana dengan sistem bantuan dan manual online? Apa jenis elemen input yang harus disertakan? Apa jenis elemen output yang harus disertakan? Pertanyaan lain yang harus dijawab terkait dengan platform di mana perangkat lunak akan digunakan. Misalnya, apakah aplikasi akan berjalan di komputer yang berdiri sendiri, apakah akan didistribusikan, atau aplikasi akan mobile? Jika diharapkan berjalan di perangkat seluler, jenis platform apa: notebook, tablet, atau ponsel? Apakah akan disebarkan menggunakan teknologi Web, yang berjalan di banyak perangkat, atau akan dibuat menggunakan aplikasi berbasis Android dari Google, iOS dari Apple, atau Windows dari Microsoft ? Tergantung pada jawaban atas pertanyaan-pertanyaan ini, berbagai jenis antarmuka pengguna dimungkinkan.

Dengan munculnya platform jejaring sosial, seperti Facebook, Twitter, blog, YouTube, dan LinkedIn, implikasi untuk antarmuka pengguna dapat membingungkan. Tergantung pada aplikasinya, platform jejaring sosial yang berbeda mungkin sesuai untuk berbagai aspek aplikasi. Selanjutnya, masing-masing platform jejaring sosial yang berbeda memungkinkan (atau mencegah) pertimbangan berbagai jenis antarmuka pengguna. Terakhir, dengan audiens potensial aplikasi Anda menjadi global, banyak masalah budaya yang berbeda akan muncul dalam desain dan pengembangan antarmuka pengguna yang sadar budaya (seperti persyaratan multibahasa). Jelas, deskripsi lengkap tentang semua masalah yang terkait dengan interaksi manusia-komputer berada di luar cakupan buku ini. Namun, dari sudut pandang pengguna, antarmuka pengguna adalah sistem. Kami menyajikan isu-isu dasar dalam desain antarmuka pengguna di Bab 10.

Arsitektur Fisik. ***Lapisan arsitektur fisik*** membahas bagaimana perangkat lunak akan dijalankan pada komputer dan jaringan tertentu. Lapisan ini mencakup kelas-kelas yang berhubungan dengan komunikasi antara perangkat lunak dan sistem operasi komputer dan jaringan. Misalnya, kelas yang membahas cara berinteraksi dengan berbagai port pada komputer tertentu termasuk dalam lapisan ini.

Tidak seperti di lapisan pondasi, banyak masalah desain harus ditangani sebelum memilih set kelas yang sesuai untuk lapisan ini. Masalah desain ini termasuk pilihan komputasi atau arsitektur jaringan (seperti berbagai arsitektur client-server), desain jaringan yang sebenarnya, spesifikasi perangkat keras dan perangkat lunak server, dan masalah keamanan. Masalah lain yang harus ditangani dengan desain lapisan ini termasuk konfigurasi perangkat keras dan perangkat lunak komputer (pilihan sistem operasi, seperti Linux, Mac OSX, dan Windows; jenis dan kecepatan prosesor; jumlah memori; penyimpanan data; dan input/output teknologi), standarisasi, virtualisasi, komputasi grid, komputasi terdistribusi, dan layanan Web. Ini kemudian membawa kita ke salah satu gorila pepatah di sudut. Apa yang Anda lakukan dengan awan? Cloud pada dasarnya adalah bentuk komputasi terdistribusi.

Dalam hal ini, cloud memungkinkan Anda memperlakukan platform, infrastruktur, perangkat lunak, dan bahkan proses bisnis sebagai layanan jarak jauh yang dapat dikelola oleh perusahaan lain. Dalam banyak hal, cloud memungkinkan sebagian besar TI di-outsource (lihat pembahasan outsourcing nanti di bab ini). Juga seperti yang diangkat dengan lapisan interaksi manusia-komputer, seluruh masalah komputasi seluler sangat relevan dengan lapisan ini. Secara khusus, perangkat yang berbeda, seperti ponsel dan tablet, relevan dan cara mereka berkomunikasi satu sama lain, seperti melalui jaringan seluler atau WiFi, juga penting.

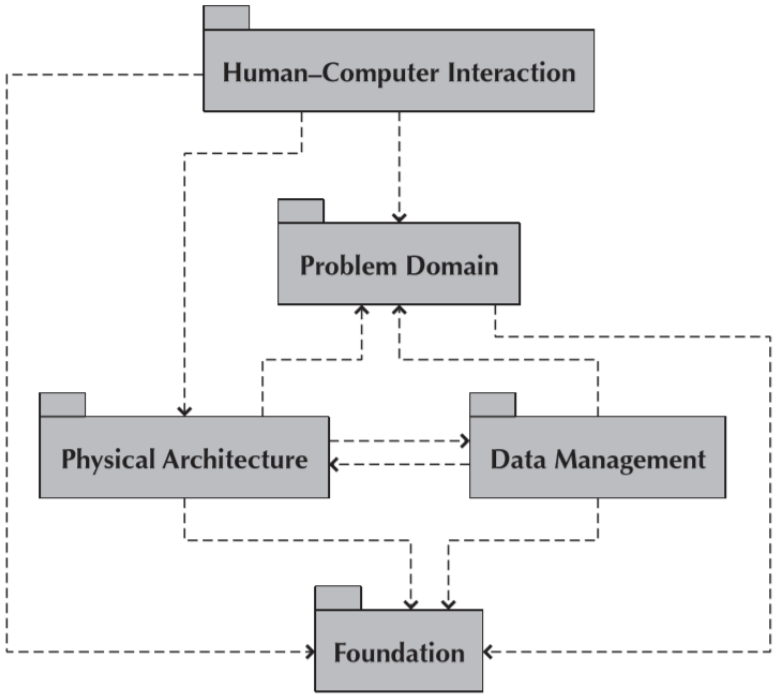
Akhirnya, mengingat besarnya daya yang dibutuhkan TI saat ini, seluruh topik TI Hijau harus dibahas. Topik yang perlu ditangani terkait dengan Green IT adalah lokasi pusat data, pendinginan pusat data, sumber daya alternatif, pengurangan bahan habis pakai, gagasan kantor tanpa kertas, kepatuhan Energy Star, dan potensi dampak virtualisasi, cloud , dan komputasi seluler. Seperti halnya manajemen data dan lapisan interaksi manusia-komputer, deskripsi lengkap tentang semua masalah yang terkait dengan arsitektur fisik berada di luar cakupan buku ini. Namun, kami menyajikan masalah dasar dalam Bab 11.

7.4 PAKET DAN DIAGRAM PAKET

Dalam UML, kolaborasi, partisi, dan lapisan dapat diwakili oleh konstruksi tingkat yang lebih tinggi: sebuah paket. Sebenarnya, sebuah paket memiliki tujuan yang sama seperti folder di komputer Anda. Ketika paket digunakan dalam bahasa pemrograman seperti Java, paket sebenarnya diimplementasikan sebagai folder. Paket adalah konstruksi umum yang dapat diterapkan ke salah satu elemen dalam model UML. Dalam Bab 4, kami memperkenalkan ide paket sebagai cara untuk mengelompokkan Use-Case bersama-sama untuk membuat diagram Use-Case lebih mudah dibaca dan untuk menjaga model pada tingkat kompleksitas yang wajar. Dalam Bab 5 dan 6, kami melakukan hal yang sama untuk diagram kelas dan komunikasi. Di bagian ini, kami menjelaskan diagram paket: diagram yang hanya terdiri dari paket. Diagram paket secara efektif merupakan diagram kelas yang hanya menampilkan paket.

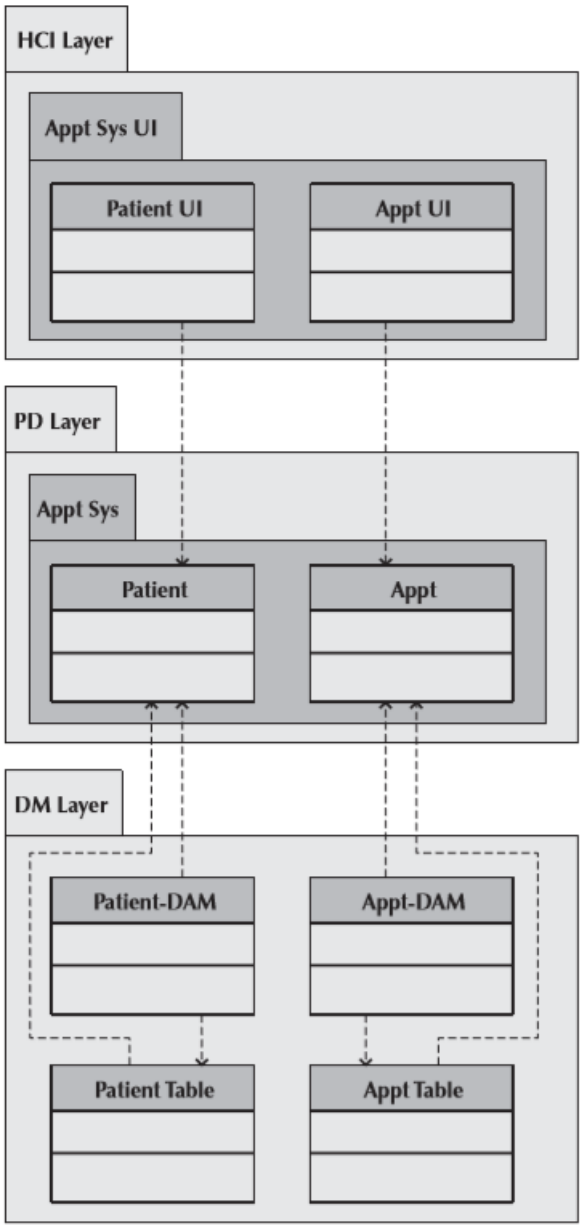
Sebuah paket: - Adalah pengelompokan logis elemen UML. - Digunakan untuk menyederhanakan diagram UML dengan mengelompokkan elemen-elemen terkait menjadi satu elemen tingkat yang lebih tinggi.	
Hubungan ketergantungan: - Mewakili ketergantungan antar paket: Jika sebuah paket diubah, paket dependen juga harus dimodifikasi. - Memiliki panah yang ditarik dari paket dependen menuju paket yang menjadi dependennya.	

Gambar 7-18 Sintaks untuk Diagram Paket



Gambar 7-19 Package Diagram Hubungan Ketergantungan Antar Lapisan

Simbol untuk sebuah paket mirip dengan folder dengan tab (lihat Gambar 7-18). Tergantung di mana sebuah paket digunakan, paket dapat berpartisipasi dalam berbagai jenis hubungan. Misalnya, dalam diagram kelas, paket mewakili pengelompokan kelas. Oleh karena itu, hubungan agregasi dan asosiasi dimungkinkan.



Gambar 7-20 Diagram Paket Parsial Sistem Penunjukan

Dalam diagram paket, berguna untuk menggambarkan hubungan baru, hubungan ketergantungan. Hubungan ketergantungan digambarkan dengan panah putus-putus (lihat Gambar 7-18). Hubungan ketergantungan mewakili fakta bahwa ada ketergantungan modifikasi antara dua paket. Artinya, ada kemungkinan perubahan dalam satu paket dapat menyebabkan perubahan diperlukan di paket lain. Gambar 7-19 menggambarkan ketergantungan antara lapisan yang berbeda (fondasi, domain masalah, manajemen data, interaksi manusia-komputer, dan arsitektur fisik). Misalnya, jika perubahan terjadi pada lapisan domain masalah, kemungkinan besar akan menyebabkan perubahan terjadi pada interaksi manusia-komputer, arsitektur fisik, dan lapisan manajemen data. Perhatikan bahwa lapisan ini menunjuk ke lapisan domain masalah dan karena itu bergantung padanya. Namun, sebaliknya tidak benar. Perhatikan juga bahwa semua lapisan bergantung pada lapisan pondasi. Ini karena isi dari lapisan pondasi menjadi kelas dasar dari mana semua kelas lain akan dibangun. Akibatnya, setiap perubahan yang dilakukan pada lapisan ini dapat berdampak pada semua lapisan lainnya.

Di tingkat kelas, mungkin ada banyak penyebab ketergantungan antar kelas. Misalnya, jika protokol untuk suatu metode diubah, maka ini menyebabkan antarmuka untuk semua objek dari kelas ini berubah. Oleh karena itu, semua kelas yang memiliki objek yang mengirim pesan ke instance kelas yang dimodifikasi mungkin harus dimodifikasi. Menangkap hubungan ketergantungan antara kelas dan paket membantu organisasi dalam memelihara sistem informasi berorientasi objek.

Kolaborasi, partisi, dan lapisan dimodelkan sebagai paket dalam UML. Kolaborasi biasanya diperhitungkan ke dalam satu set partisi, yang biasanya ditempatkan pada lapisan. Partisi dapat terdiri dari partisi lain. Juga, dimungkinkan untuk memiliki kelas di partisi, yang terdapat di partisi lain, yang ditempatkan pada lapisan. Semua pengelompokan ini diwakili menggunakan paket dalam UML. Ingat bahwa paket hanyalah konstruksi pengelompokan generik yang digunakan untuk menyederhanakan model UML melalui penggunaan komposisi.

Diagram paket sederhana, berdasarkan contoh sistem penunjukan dari bab-bab sebelumnya, ditunjukkan pada Gambar 7-20. Diagram ini hanya menggambarkan sebagian kecil dari keseluruhan sistem. Dalam hal ini, kita melihat bahwa kelas Pasien UI, Pasien-DAM, dan Tabel Pasien bergantung pada kelas Pasien. Selanjutnya, kelas Pasien-DAM tergantung pada kelas Tabel Pasien. Hal yang sama dapat dilihat dengan kelas yang berhubungan dengan janji yang sebenarnya. Dengan mengisolasi kelas Domain Masalah (seperti kelas Patient dan Appt) dari kelas objectpersistence yang sebenarnya (seperti kelas Patient Table dan Appt Table) melalui penggunaan kelas Manajemen Data perantara (kelas Patient-DAM dan Appt-DAM) , kami mengisolasi kelas Domain Masalah dari media penyimpanan yang sebenarnya. Ini sangat menyederhanakan pemeliharaan dan meningkatkan penggunaan kembali kelas Domain Masalah. Tentu saja, dalam deskripsi lengkap dari sistem nyata, akan ada lebih banyak dependensi.

Panduan untuk Membuat Diagram Paket

Seperti diagram UML yang dijelaskan dalam bab-bab sebelumnya, kami menyediakan seperangkat pedoman yang telah kami adaptasi dari Ambler untuk membuat diagram paket. Dalam hal ini, kami menawarkan enam pedoman.

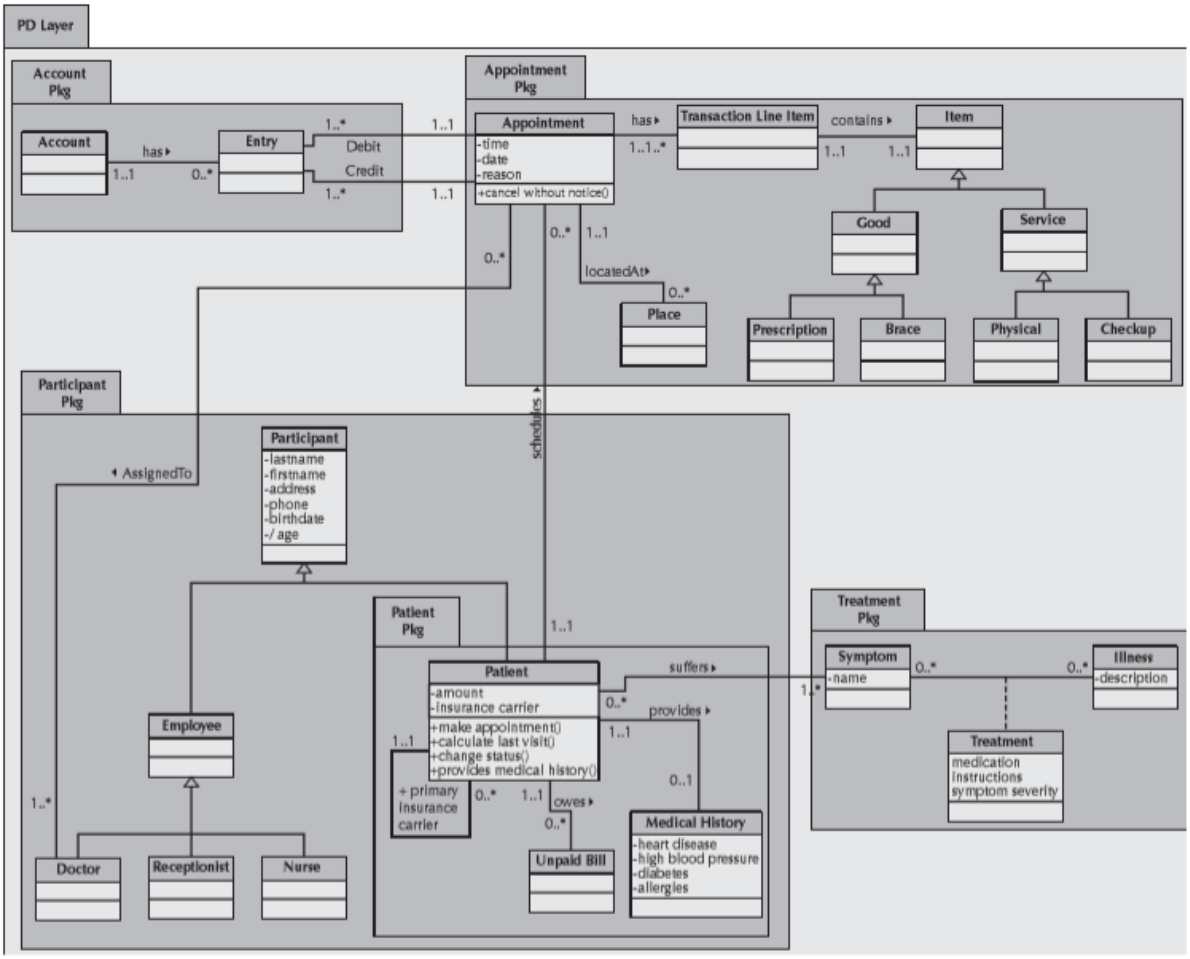
- Gunakan diagram paket untuk mengatur desain secara logis. Secara khusus, gunakan paket untuk mengelompokkan kelas bersama-sama ketika ada hubungan pewarisan, agregasi, atau komposisi di antara mereka atau ketika kelas membentuk kolaborasi.
- Dalam beberapa kasus, ada hubungan pewarisan, agregasi, atau asosiasi antar paket. Dalam kasus tersebut, untuk tujuan keterbacaan, cobalah untuk mendukung hubungan pewarisan secara vertikal, dengan paket yang berisi superclass ditempatkan di atas paket yang berisi subclass. Gunakan penempatan horizontal untuk mendukung hubungan agregasi dan asosiasi, dengan paket ditempatkan berdampingan.
- Ketika hubungan ketergantungan ada pada diagram, itu menyiratkan bahwa setidaknya ada satu hubungan semantik antara elemen dari dua paket. Arah ketergantungan biasanya dari subclass ke superclass, dari keseluruhan ke bagian, dan dengan kontrak, dari klien ke server. Dengan kata lain, subclass bergantung pada keberadaan superclass, keseluruhan bergantung pada bagian-bagiannya yang ada, dan klien tidak dapat mengirim pesan ke server yang tidak ada.
- Saat menggunakan paket untuk mengelompokkan Use-Case bersama-sama, pastikan untuk menyertakan aktor dan asosiasi yang mereka miliki dengan Use-Case yang dikelompokkan dalam paket. Ini akan memungkinkan pengguna diagram untuk lebih memahami konteks diagram.
- Berikan setiap paket nama yang sederhana namun deskriptif untuk memberikan informasi yang cukup kepada pengguna diagram paket untuk memahami apa yang

dienkapsulasi oleh paket. Jika tidak, pengguna harus menelusuri atau membuka paket untuk memahami tujuan paket.

- Pastikan bahwa paket-paket itu kohesif. Agar sebuah paket menjadi kohesif, kelas-kelas yang terkandung dalam paket tersebut, dalam beberapa hal, menjadi milik bersama. Aturan sederhana, tetapi tidak sempurna, untuk diikuti ketika mengelompokkan kelas bersama dalam sebuah paket adalah bahwa semakin banyak kelas bergantung satu sama lain, semakin besar kemungkinan mereka menjadi satu dalam sebuah paket.

Membuat Diagram Paket

Di bagian ini, kami menjelaskan proses lima langkah sederhana untuk membuat diagram paket. Langkah pertama adalah mengatur konteks untuk diagram paket. Ingat, paket dapat digunakan untuk memodelkan partisi dan/atau lapisan. Meninjau kembali sistem janji temu, mari kita atur konteksnya sebagai lapisan domain masalah.

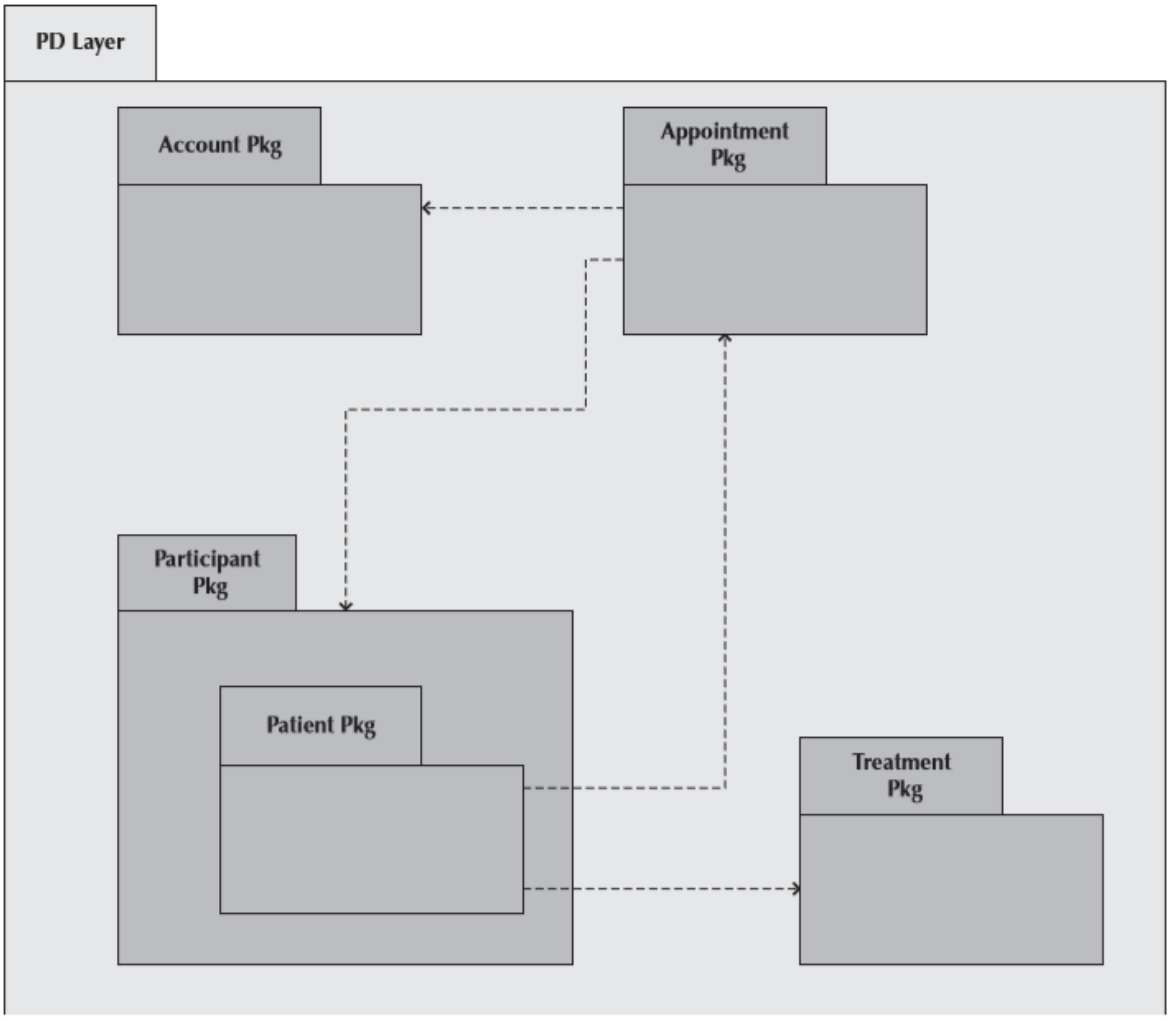


Gambar 7-21 Diagram Paket Lapisan PD Masalah Pengangkatan

Langkah kedua adalah mengelompokkan kelas-kelas menjadi partisi berdasarkan hubungan yang dimiliki kelas-kelas tersebut. Hubungan tersebut meliputi generalisasi, agregasi, berbagai asosiasi, dan pengiriman pesan yang terjadi antara objek dalam sistem. Untuk mengidentifikasi paket dalam sistem janji temu, kita harus melihat model analisis yang berbeda [misalnya, diagram kelas (lihat Gambar 7-15), diagram komunikasi (lihat Gambar 7-10)], dan matriks CRUDE (lihat Gambar 7-12). Kelas dalam hierarki generalisasi harus disimpan bersama dalam satu partisi.

Langkah ketiga adalah menempatkan kelas-kelas yang dikelompokkan bersama dalam sebuah partisi dan memodelkan partisi-partisi tersebut sebagai paket. Gambar 7-21 menggambarkan lima paket di Lapisan PD: Paket Akun, Paket Peserta, Paket Pasien, Paket Janji Temu, dan Paket Perawatan.

Langkah keempat adalah mengidentifikasi hubungan ketergantungan antar paket. Kami mencapai ini dengan meninjau hubungan yang melintasi batas paket untuk mengungkap dependensi potensial. Dalam sistem janji temu, kita melihat hubungan asosiasi yang menghubungkan Paket Akun dengan Paket Pengangkatan (melalui asosiasi antara kelas Masuk dan kelas Pengangkatan), Paket Peserta dengan Paket Pengangkatan (melalui asosiasi antara kelas Dokter dan Kelas Pengangkatan kelas), Paket Pasien, yang terdapat dalam Paket Peserta, dengan Paket Pengangkatan (melalui asosiasi antara kelas Pasien dan Pengangkatan), dan Paket Pasien dengan Paket Perawatan (melalui asosiasi antara kelas Pasien dan Gejala).



Gambar 7-22 Ikhtisar Package Diagram Lapisan PD untuk Sistem Appointment

Langkah kelima adalah layoutdan menggambar diagram. Dengan menggunakan pedoman, tempatkan paket dan hubungan ketergantungan dalam diagram. Dalam hal sistem Penunjukan, terdapat hubungan ketergantungan antara Paket Akun dan Paket Penunjukan, Paket Peserta dan Paket Penunjukan, Paket Pasien dan Paket Penunjukan, dan Paket Pasien dan Paket Perawatan. Untuk meningkatkan pemahaman hubungan ketergantungan antar paket yang berbeda, diagram paket murni yang hanya menunjukkan hubungan ketergantungan antar paket dapat dibuat (lihat Gambar 7-22).

Verifikasi dan Validasi Diagram Paket

Seperti semua model sebelumnya, diagram paket perlu diverifikasi dan divalidasi. Dalam hal ini, diagram paket diturunkan terutama dari diagram kelas, diagram komunikasi, dan matriks CRUDE. Hanya dua area yang perlu ditinjau.

Pertama, paket yang diidentifikasi harus masuk akal dari sudut pandang domain masalah. Misalnya, dalam konteks sistem janji temu, paket pada Gambar 7-22 (Peserta, Pasien, Aplikasi, Akun, dan Perawatan) tampaknya masuk akal.

Kedua, semua hubungan ketergantungan harus didasarkan pada hubungan pengiriman pesan pada diagram komunikasi, entri sel dalam matriks CRUDE, dan asosiasi pada diagram kelas. Dalam hal sistem penunjukan, hubungan ketergantungan yang teridentifikasi adalah wajar (lihat Gambar 7-10, 7-12, 7-15, dan 7-22).

7.5 STRATEGI DESAIN

Sampai saat ini, kami berasumsi bahwa sistem akan dibangun dan diimplementasikan oleh tim proyek; namun, sebenarnya ada tiga cara untuk mendekati pembuatan sistem baru: mengembangkan aplikasi kustom in-house, membeli dan menyesuaikan sistem yang dikemas, dan mengandalkan vendor eksternal, pengembang, atau penyedia layanan untuk membangun sistem. Masing-masing pilihan ini memiliki kekuatan dan kelemahan, dan masing-masing lebih sesuai dalam skenario yang berbeda. Bagian berikut menjelaskan setiap pilihan desain secara bergantian, dan kemudian kami menyajikan kriteria yang dapat Anda gunakan untuk memilih salah satu dari tiga pendekatan untuk proyek Anda.

Pengembangan Kustom

Banyak tim proyek berasumsi bahwa pengembangan kustom, atau membangun sistem baru dari awal, adalah cara terbaik untuk membuat sistem. Untuk satu hal, tim memiliki kendali penuh atas tampilan dan fungsi sistem. Pengembangan kustom juga memungkinkan pengembang untuk menjadi fleksibel dan kreatif dalam memecahkan masalah bisnis. Selain itu, aplikasi kustom lebih mudah diubah untuk menyertakan komponen yang memanfaatkan teknologi saat ini yang dapat mendukung upaya strategis tersebut.

Membangun sistem in-house juga membangun keterampilan teknis dan pengetahuan fungsional di dalam perusahaan. Saat pengembang bekerja dengan pengguna bisnis, pemahaman mereka tentang bisnis tumbuh dan mereka menjadi lebih mampu menyelaraskan SI dengan strategi dan kebutuhan. Pengembang yang sama ini mendaki kurva pembelajaran teknologi sehingga proyek masa depan yang menerapkan teknologi serupa membutuhkan lebih sedikit usaha.

Pengembangan aplikasi kustom, bagaimanapun, membutuhkan upaya khusus yang melibatkan berjam-jam dan kerja keras. Banyak perusahaan memiliki staf pengembangan yang sudah terlalu berkomitmen untuk memenuhi backlog besar permintaan sistem dan tidak punya waktu untuk proyek lain. Selain itu, berbagai keterampilan—teknis, interpersonal, fungsional, manajemen proyek, dan pemodelan—harus tersedia agar proyek dapat berjalan dengan lancar. Profesional IS, terutama individu yang sangat terampil, cukup sulit untuk dipekerjakan dan dipertahankan.

Risiko yang terkait dengan membangun sistem dari bawah ke atas bisa sangat tinggi, dan tidak ada jaminan bahwa proyek tersebut akan berhasil. Pengembang dapat ditarik untuk

mengerjakan proyek lain, hambatan teknis dapat menyebabkan penundaan yang tidak terduga, dan pengguna bisnis dapat menjadi tidak sabar dengan garis waktu yang terus bertambah.

Paket Software

Banyak kebutuhan bisnis yang tidak unik, dan karena tidak masuk akal untuk menemukan kembali roda, banyak organisasi membeli paket perangkat lunak yang telah ditulis daripada mengembangkan solusi kustom mereka sendiri. Faktanya, ada ribuan program perangkat lunak yang tersedia secara komersial yang telah ditulis untuk melayani banyak tujuan. Pikirkan tentang kebutuhan Anda sendiri akan pengolah kata—apakah Anda pernah mempertimbangkan untuk menulis perangkat lunak pengolah kata Anda sendiri? Itu akan sangat konyol mengingat jumlah paket perangkat lunak bagus yang tersedia yang relatif murah.

Demikian pula, sebagian besar perusahaan memiliki kebutuhan yang dapat dipenuhi dengan cukup baik oleh perangkat lunak yang dikemas, seperti penggajian atau piutang. Akan jauh lebih efisien untuk membeli program yang telah dibuat, diuji, dan terbukti. Selain itu, sistem yang dikemas dapat dibeli dan dipasang dalam waktu yang relatif singkat jika dibandingkan dengan sistem kustom. Plus, sistem paket menggabungkan keahlian dan pengalaman vendor yang menciptakan perangkat lunak.

Perangkat lunak yang dikemas dapat berkisar dari komponen yang dapat digunakan kembali hingga alat kecil, satu fungsi hingga sistem besar yang mencakup semua seperti aplikasi perencanaan sumber daya perusahaan (ERP) yang diinstal untuk mengotomatisasi seluruh bisnis. Menerapkan sistem ERP adalah proses di mana organisasi besar menghabiskan jutaan dolar untuk menginstal paket oleh perusahaan seperti SAP atau Oracle dan kemudian mengubah bisnis mereka sesuai dengan itu. Menginstal perangkat lunak ERP jauh lebih sulit daripada menginstal paket aplikasi kecil karena manfaatnya bisa lebih sulit untuk direalisasikan dan masalahnya jauh lebih serius.

Namun, ada masalah yang terkait dengan paket perangkat lunak. Misalnya, perusahaan yang membeli sistem paket harus menerima fungsionalitas yang disediakan oleh sistem, dan jarang ada yang cocok. Jika sistem paket memiliki cakupan yang besar, implementasinya dapat berarti perubahan substansial dalam cara perusahaan melakukan bisnis. Membiarkan teknologi mendorong bisnis bisa berbahaya.

Sebagian besar aplikasi yang dikemas memungkinkan kustomisasi, atau manipulasi parameter sistem untuk mengubah cara kerja fitur tertentu. Misalnya, paket mungkin memiliki cara untuk menerima informasi tentang perusahaan Anda atau logo perusahaan yang kemudian akan muncul di layar input. Atau paket perangkat lunak akuntansi dapat menawarkan pilihan berbagai cara untuk menangani arus kas atau pengendalian persediaan sehingga dapat mendukung praktik akuntansi di berbagai organisasi. Jika jumlah kustomisasi tidak cukup dan paket perangkat lunak memiliki beberapa fitur yang tidak berfungsi seperti yang dibutuhkan perusahaan untuk bekerja, tim proyek dapat membuat solusi.

Solusi adalah program tambahan yang dibuat khusus yang berinteraksi dengan aplikasi yang dikemas untuk menangani kebutuhan khusus. Ini bisa menjadi cara yang bagus untuk membuat fungsionalitas yang dibutuhkan yang tidak ada dalam paket perangkat lunak. Tetapi solusi harus menjadi pilihan terakhir karena beberapa alasan. Pertama, solusi tidak didukung oleh vendor yang menyediakan paket perangkat lunak, sehingga upgrade ke sistem utama mungkin membuat solusi tidak efektif. Juga, jika masalah muncul, vendor cenderung menyalahkan solusi sebagai penyebab dan menolak untuk memberikan dukungan.

Meskipun memilih sistem perangkat lunak paket lebih sederhana daripada pengembangan kustom, itu juga dapat mengambil manfaat dari mengikuti metodologi formal, seperti jika aplikasi kustom sedang dibangun.

Integrasi sistem mengacu pada proses membangun sistem baru dengan menggabungkan perangkat lunak yang dikemas, sistem warisan yang ada, dan perangkat lunak baru yang ditulis untuk mengintegrasikannya. Banyak perusahaan konsultan mengkhususkan diri dalam integrasi sistem, sehingga tidak jarang perusahaan memilih opsi paket perangkat lunak dan kemudian mengalihdayakan integrasi berbagai paket ke perusahaan konsultan. (Outsourcing dibahas di bagian berikutnya.)

Tantangan utama dalam integrasi sistem adalah menemukan cara untuk mengintegrasikan data yang dihasilkan oleh berbagai paket dan sistem lama. Integrasi sering bergantung pada pengambilan data yang dihasilkan oleh satu paket atau sistem dan memformat ulang untuk digunakan dalam paket atau sistem lain. Tim proyek mulai dengan memeriksa data yang dihasilkan dan dibutuhkan oleh paket atau sistem yang berbeda dan mengidentifikasi transformasi yang harus terjadi untuk memindahkan data dari satu ke yang lain. Dalam banyak kasus, ini melibatkan membodohi paket atau sistem yang berbeda dengan berpikir bahwa data dihasilkan oleh modul program yang ada yang diharapkan paket atau sistem untuk menghasilkan data daripada paket atau sistem baru yang sedang terintegrasi.

Pendekatan ketiga adalah melalui penggunaan pembungkus objek. Pembungkus objek pada dasarnya adalah objek yang "membungkus" sistem warisan, memungkinkan sistem berorientasi objek untuk mengirim pesan ke sistem warisan. Secara efektif, pembungkus objek membuat antarmuka program aplikasi (API) ke sistem lama. Pembuatan pembungkus objek melindungi investasi perusahaan dalam sistem warisan.

Outsourching

Pilihan desain yang membutuhkan paling sedikit sumber daya internal adalah outsourcing—mempekerjakan vendor eksternal, pengembang, atau penyedia layanan untuk membuat sistem. Outsourcing telah menjadi sangat populer dalam beberapa tahun terakhir. Beberapa memperkirakan bahwa sebanyak 50 persen perusahaan dengan anggaran TI lebih dari Rp. 7,5 miliar saat ini melakukan outsourcing atau mengevaluasi pendekatan tersebut.

Dengan outsourcing, pengambilan keputusan dan/atau kontrol manajemen dari fungsi bisnis dialihkan ke pemasok luar. Transfer ini membutuhkan koordinasi dua arah, pertukaran informasi, dan kepercayaan antara pemasok dan bisnis. Dari perspektif TI, outsourcing TI dapat mencakup menyewa konsultan untuk memecahkan masalah tertentu, mempekerjakan programmer kontrak untuk mengimplementasikan solusi, mempekerjakan perusahaan untuk mengelola fungsi dan aset TI perusahaan, atau benar-benar mengalihdayakan seluruh fungsi TI ke perusahaan yang terpisah. . Saat ini, melalui penggunaan penyedia layanan aplikasi (ASP), teknologi layanan Web, dan layanan cloud, adalah mungkin untuk menggunakan pendekatan bayar sesuai penggunaan untuk paket perangkat lunak. Pada dasarnya, outsourcing TI melibatkan perekrutan pihak ketiga untuk melakukan beberapa fungsi TI yang secara tradisional akan dilakukan di rumah.

Ada manfaat besar untuk meminta orang lain mengembangkan sistem perusahaan. Perusahaan luar mungkin lebih berpengalaman dalam teknologi atau memiliki lebih banyak sumber daya, seperti programmer berpengalaman. Banyak perusahaan memulai kesepakatan outsourcing untuk mengurangi biaya, sedangkan yang lain melihatnya sebagai peluang untuk menambah nilai bisnis.

Untuk alasan apapun, outsourcing dapat menjadi alternatif yang baik untuk sistem baru. Namun, itu tidak datang tanpa biaya. Jika Anda memutuskan untuk menyerahkan pembuatan sistem baru di tangan orang lain, Anda dapat membahayakan informasi rahasia atau kehilangan kendali atas pengembangan di masa mendatang. Profesional internal tidak mendapatkan manfaat dari keterampilan yang dapat dipelajari dari proyek; alih-alih, keahlian dipindahkan ke organisasi luar. Pada akhirnya, keterampilan penting dapat berjalan keluar dari pintu di akhir kontrak. Selain itu, ketika outsourcing lepas pantai dipertimbangkan, kita juga harus menyadari masalah bahasa, perbedaan zona waktu, dan perbedaan budaya (misalnya, praktik bisnis yang dapat diterima sebagaimana dipahami di satu negara yang mungkin tidak dapat diterima di negara lain). Semua masalah ini, jika tidak ditangani dengan benar, dapat mengalahkan keuntungan apa pun yang dapat direalisasikan oleh outsourcing atau outsourcing lepas pantai.

Sebagian besar risiko dapat diatasi jika perusahaan memutuskan untuk melakukan outsourcing, tetapi dua sangat penting. Pertama, perusahaan harus benar-benar menilai persyaratan untuk proyek—perusahaan tidak boleh mengalihdayakan apa yang tidak dipahami. Jika perencanaan dan analisis yang ketat telah terjadi, maka perusahaan harus menyadari kebutuhannya dengan baik. Kedua, perusahaan harus hati-hati memilih vendor, pengembang, atau layanan dengan rekam jejak yang terbukti dengan jenis sistem dan teknologi yang dibutuhkan sistemnya.

Tiga jenis utama kontrak dapat ditarik untuk mengontrol kesepakatan outsourcing. Kontrak waktu dan pengaturan sangat fleksibel karena perusahaan setuju untuk membayar berapa pun waktu dan biaya yang diperlukan untuk menyelesaikan pekerjaan. Tentu saja, kesepakatan ini bisa menghasilkan tagihan besar yang melebihi perkiraan awal. Ini bekerja paling baik ketika perusahaan dan agen outsourcing tidak jelas tentang apa yang akan dilakukan untuk menyelesaikan pekerjaan.

Sebuah perusahaan akan membayar tidak lebih dari yang diharapkan dengan kontrak harga tetap karena jika outsourcing melebihi harga yang disepakati, ia harus menyerap biaya. Agen outsourcing jauh lebih berhati-hati dalam mendefinisikan persyaratan dengan jelas di awal, dan hanya ada sedikit fleksibilitas untuk perubahan.

Jenis kontrak yang semakin populer adalah kontrak nilai tambah, di mana agen outsourcing menuai beberapa persentase dari manfaat sistem yang telah selesai. Perusahaan memiliki risiko yang sangat kecil dalam kasus ini, tetapi harus berharap untuk berbagi kekayaan begitu sistem diterapkan.

Membuat kontrak yang adil adalah seni karena fleksibilitas harus diimbangi dengan hati-hati dengan persyaratan yang jelas. Seringkali, kebutuhan berubah seiring waktu. Oleh karena itu, kontrak tidak boleh terlalu spesifik dan kaku sehingga perubahan tidak dapat dilakukan. Pikirkan tentang seberapa cepat teknologi seluler telah berubah. Sulit untuk meramalkan bagaimana sebuah proyek dapat berkembang dalam jangka waktu yang lama. Kontrak jangka pendek membantu memberikan ruang untuk penilaian ulang jika kebutuhan berubah atau jika hubungan tidak berjalan seperti yang diharapkan kedua belah pihak. Dalam semua kasus, hubungan dengan agen outsourcing harus dilihat sebagai kemitraan di mana kedua belah pihak diuntungkan dan berkomunikasi secara terbuka.

Mengelola hubungan outsourcing adalah pekerjaan penuh waktu. Dengan demikian, seseorang perlu ditugaskan penuh waktu untuk mengelola agen outsourcing, dan tingkat orang tersebut harus sesuai dengan ukuran pekerjaan (pertunangan outsourcing jutaan dolar harus ditangani oleh eksekutif tingkat tinggi). Sepanjang hubungan, kemajuan harus dilacak dan diukur terhadap tujuan yang telah ditentukan. Jika sebuah perusahaan memulai strategi desain outsourcing, harus dipastikan untuk mendapatkan informasi yang memadai. Banyak buku

telah ditulis yang memberikan informasi lebih rinci tentang topik tersebut. Gambar 7-23 merangkum beberapa pedoman untuk outsourcing.

Outsourcing
• Jaga agar jalur komunikasi tetap terbuka antara Anda dan agen outsourcing Anda. • Tetapkan dan stabilkan persyaratan sebelum menandatangani kontak. • Lihat hubungan outsourcing sebagai kemitraan. • Pilih vendor, pengembang, atau penyedia layanan dengan hati-hati. • Tetapkan seseorang untuk mengelola hubungan. • Jangan mengalihdayakan apa yang tidak Anda pahami. • Tekankan persyaratan fleksibel, hubungan jangka panjang dan kontrak jangka pendek.

Gambar 7-23 Pedoman Outsourching

Memilih Strategi Desain

Masing-masing strategi desain yang baru saja dibahas memiliki kekuatan dan kelemahannya, dan tidak ada satu strategi yang secara inheren lebih baik dari yang lain. Oleh karena itu, penting untuk memahami kekuatan dan kelemahan masing-masing strategi dan kapan harus menggunakannya. Gambar 7-24 merangkum karakteristik masing-masing strategi.

Kebutuhan Bisnis. Jika kebutuhan bisnis untuk sistem sudah umum dan solusi teknis sudah ada yang dapat memenuhi kebutuhan bisnis sistem, tidak masuk akal untuk membangun aplikasi kustom. Sistem yang dikemas adalah alternatif yang baik untuk kebutuhan bisnis umum. Alternatif kustom harus dieksplorasi ketika kebutuhan bisnis unik atau memiliki persyaratan khusus. Biasanya, jika kebutuhan bisnis tidak terlalu penting bagi perusahaan, maka outsourcing adalah pilihan terbaik—seseorang di luar organisasi dapat bertanggung jawab atas pengembangan aplikasi.

Pengalaman di rumah. Jika ada pengalaman internal untuk semua kebutuhan fungsional dan teknis sistem, akan lebih mudah untuk membangun aplikasi kustom daripada jika keterampilan ini tidak ada. Sistem yang dikemas dapat menjadi alternatif yang lebih baik bagi perusahaan yang tidak memiliki keterampilan teknis untuk membangun sistem yang diinginkan. Misalnya, tim proyek yang tidak memiliki keterampilan teknologi seluler mungkin ingin mempertimbangkan untuk mengalihdayakan aspek-aspek sistem tersebut.

Keterampilan Proyek. Keterampilan yang diterapkan selama proyek baik teknis (misalnya, Java, SQL) atau fungsional (misalnya, keamanan), dan alternatif desain yang berbeda lebih layak, tergantung pada seberapa penting keterampilan untuk strategi perusahaan. Misalnya, jika keahlian fungsional dan teknis tertentu yang terkait dengan pengembangan aplikasi seluler penting bagi organisasi karena mengharapkan seluler memainkan peran penting dalam penjualannya dari waktu ke waktu, maka masuk akal bagi perusahaan untuk mengembangkan aplikasi seluler secara internal, menggunakan karyawan perusahaan sehingga keterampilan dapat dikembangkan dan ditingkatkan. Di sisi lain, beberapa keterampilan, seperti keamanan jaringan, mungkin berada di luar keahlian teknis karyawan atau tidak menarik bagi para ahli strategi perusahaan—ini hanya masalah operasional yang

perlu ditangani. Dalam hal ini, sistem paket atau outsourcing harus dipertimbangkan sehingga karyawan internal dapat fokus pada aplikasi dan keterampilan penting bisnis lainnya.

Manajemen proyek. Aplikasi khusus memerlukan manajemen proyek yang sangat baik dan metodologi yang telah terbukti. Begitu banyak hal, seperti hambatan pendanaan, penahanan staf, dan pengguna bisnis yang terlalu menuntut, dapat mendorong proyek keluar jalur. Oleh karena itu, tim proyek harus memilih untuk mengembangkan aplikasi kustom hanya jika dipastikan bahwa mekanisme koordinasi dan kontrol yang mendasari akan ada. Alternatif yang dikemas dan outsourcing juga perlu dikelola; namun, mereka lebih terlindungi dari hambatan internal karena pihak eksternal memiliki tujuan dan prioritas mereka sendiri (misalnya, mungkin lebih mudah bagi kontraktor luar untuk mengatakan tidak kepada pengguna daripada orang di dalam perusahaan). Biasanya, alternatif paket dan outsourcing memiliki metodologi sendiri, yang dapat menguntungkan perusahaan yang tidak memiliki metodologi yang tepat untuk digunakan.

Jangka waktu. Ketika waktu menjadi faktor, tim proyek mungkin harus mulai mencari sistem yang sudah dibangun dan diuji. Dengan cara ini, perusahaan akan memiliki gambaran yang baik tentang berapa lama paket akan ditempatkan dan apa hasil akhirnya. Kerangka waktu untuk aplikasi khusus sulit ditentukan, terutama ketika Anda mempertimbangkan berapa banyak proyek yang akhirnya kehilangan tenggat waktu penting. Jika perusahaan harus memilih alternatif pengembangan kustom dan kerangka waktunya sangat singkat, perusahaan harus mempertimbangkan untuk menggunakan teknik seperti timeboxing untuk mengelola masalah ini. Waktu untuk menghasilkan suatu sistem dengan menggunakan outsourcing sangat bergantung pada sistem dan sumber daya yang dimiliki oleh pihak outsourcing. Jika penyedia layanan memiliki layanan yang dapat digunakan untuk mendukung kebutuhan perusahaan, maka kebutuhan bisnis dapat diimplementasikan dengan cepat. Jika tidak, solusi outsourcing bisa memakan waktu selama inisiatif pengembangan kustom.

	Gunakan Pengembangan Kustom Saat...	**Gunakan Sistem Terpaket Saat…**	**Gunakan Outsourcing Saat…**
Kebutuhan Bisnis	Kebutuhan bisnis itu unik.	Kebutuhan bisnis adalah hal biasa.	Kebutuhan bisnis bukanlah inti dari bisnis.
Pengalaman di rumah	Pengalaman fungsional dan teknis internal ada	Pengalaman fungsional internal ada.	Pengalaman fungsional atau teknis internal tidak ada.
Keterampilan Proyek	Ada keinginan untuk membangun keterampilan internal.	Keterampilan tidak strategis.	Keputusan untuk melakukan outsourcing merupakan keputusan strategis.
Manajemen proyek	Proyek ini memiliki manajer proyek yang sangat terampil dan metodologi yang telah terbukti.	Proyek ini memiliki manajer proyek yang dapat mengoordinasikan upaya vendor.	Proyek ini memiliki manajer proyek yang sangat terampil di tingkat organisasi yang sesuai dengan ruang lingkup kesepakatan outsourcing.
Jangka waktu	Kerangka waktunya fleksibel.	Jangka waktunya singkat.	Jangka waktunya pendek atau fleksibel.

Gambar 7-24 Memilih Strategi Desain

7.6 MEMILIH STRATEGI AKUISISI

Setelah tim proyek memiliki pemahaman yang baik tentang seberapa baik setiap strategi desain sesuai dengan kebutuhan proyek, tim harus mulai memahami dengan tepat bagaimana menerapkan strategi ini. Misalnya, alat dan teknologi apa yang akan digunakan jika alternatif khusus dipilih? Vendor apa yang membuat sistem paket yang memenuhi kebutuhan proyek? Penyedia layanan apa yang dapat membangun sistem ini jika aplikasi di-outsource? Informasi ini dapat diperoleh dari orang-orang yang bekerja di departemen SI dan dari rekomendasi oleh pengguna bisnis. Atau, tim proyek dapat menghubungi perusahaan lain dengan kebutuhan serupa dan menyelidiki jenis sistem yang telah mereka terapkan. Vendor dan konsultan biasanya bersedia memberikan informasi tentang berbagai alat dan solusi dalam bentuk brosur, demonstrasi produk, dan seminar informasi. Namun, perusahaan harus memastikan untuk memvalidasi informasi yang diterimanya dari vendor dan konsultan. Setelah semua, mereka mencoba untuk membuat penjualan. Oleh karena itu, mereka dapat memperluas kemampuan alat mereka dengan hanya berfokus pada aspek positif alat sambil menghilangkan kelemahan alat.

Kemungkinan tim proyek akan mengidentifikasi beberapa cara agar sistem dapat dibangun setelah mempertimbangkan pilihan desain yang spesifik. Misalnya, tim proyek mungkin telah menemukan tiga vendor yang membuat sistem paket yang berpotensi memenuhi kebutuhan proyek. Atau tim mungkin memperdebatkan apakah akan mengembangkan sistem menggunakan Java sebagai alat pengembangan dan sistem manajemen database dari Oracle atau mengalihdayakan upaya pengembangan ke perusahaan konsultan seperti Accenture atau CGI. Setiap alternatif memiliki pro dan kontra yang terkait dengannya yang perlu dipertimbangkan, dan hanya satu solusi yang dapat dipilih pada akhirnya.

Untuk membantu dalam keputusan ini, informasi tambahan harus dikumpulkan. Tim proyek menggunakan beberapa pendekatan untuk mengumpulkan informasi tambahan yang diperlukan. Salah satu alat yang berguna adalah request for proposal (RFP), sebuah dokumen yang meminta proposal resmi dari vendor, pengembang, atau penyedia layanan potensial. RFP menjelaskan secara rinci sistem atau layanan yang dibutuhkan, dan vendor merespons dengan menjelaskan secara rinci bagaimana mereka dapat memasok kebutuhan tersebut.

Meskipun tidak ada cara standar untuk menulis RFP, RFP harus mencakup fakta-fakta utama tertentu yang diperlukan vendor, seperti deskripsi kebutuhan yang terperinci, kebutuhan atau keadaan teknis khusus, kriteria evaluasi, prosedur yang harus diikuti, dan jadwal. Dalam proyek besar, RFP bisa mencapai ratusan halaman, karena semua detail proyek yang diperlukan harus disertakan.

RFP bukan hanya cara untuk mengumpulkan informasi. Sebaliknya, ini menghasilkan proposal vendor yang merupakan tawaran mengikat untuk menyelesaikan tugas yang dijelaskan dalam RFP. Proposal vendor mencakup jadwal dan harga untuk pekerjaan yang akan dilakukan. Setelah proposal vendor pemenang dipilih, kontrak untuk pekerjaan dikembangkan dan ditandatangani oleh kedua belah pihak.

Untuk proyek yang lebih kecil dengan anggaran yang lebih kecil, permintaan informasi (RFI) mungkin cukup. RFI adalah permintaan yang lebih pendek dan kurang rinci yang dikirim ke vendor potensial untuk mendapatkan informasi umum tentang produk dan layanan mereka. Terkadang, RFI digunakan untuk menentukan vendor mana yang memiliki kemampuan untuk melakukan layanan. Hal ini sering kemudian ditindaklanjuti dengan RFP ke vendor yang memenuhi syarat.

Ketika daftar peralatan begitu lengkap sehingga vendor hanya perlu memberikan harga, tanpa analisis atau deskripsi apa pun yang dibutuhkan, permintaan penawaran (RFQ) dapat digunakan. Misalnya, jika dua puluh pembaca tag RFID jarak jauh diperlukan dari produsen pada tanggal tertentu di lokasi tertentu, RFQ dapat digunakan. Jika suatu item dijelaskan, tetapi produk pabrikan tertentu tidak disebutkan namanya, maka pengujian ekstensif akan diperlukan untuk memverifikasi pemenuhan spesifikasi.

Matriks Alternatif

Matriks alternatif dapat digunakan untuk mengatur pro dan kontra dari alternatif desain sehingga pada akhirnya akan dipilih solusi terbaik (lihat Gambar 7-25). Matriks ini dibuat dengan langkah-langkah yang sama seperti analisis kelayakan yang disajikan pada Bab 2. Perbedaannya hanya pada matriks alternatif yang menggabungkan beberapa analisis kelayakan menjadi satu matriks sehingga alternatif dapat dengan mudah dibandingkan. Matriks alternatif adalah kisi yang berisi kelayakan teknis, anggaran, dan organisasi untuk setiap kandidat sistem, pro dan kontra yang terkait dengan mengadopsi setiap solusi, dan informasi lain yang berguna saat membuat perbandingan. Terkadang bobot disediakan untuk bagian matriks yang berbeda untuk menunjukkan kapan beberapa kriteria lebih penting untuk keputusan akhir.

Untuk membuat matriks alternatif, gambarlah kisi-kisi dengan alternatif di bagian atas dan kriteria yang berbeda (misalnya, kelayakan, pro, kontra, dan kriteria lain-lain) di sepanjang sisinya. Selanjutnya, isi kisi-kisi dengan deskripsi rinci tentang setiap alternatif. Ini menjadi dokumen yang berguna untuk diskusi karena dengan jelas menyajikan alternatif yang ditinjau dan karakteristik yang sebanding untuk masing-masing alternatif.

Terkadang, bobot dan skor ditambahkan ke matriks alternatif untuk membuat matriks alternatif berbobot yang mengomunikasikan kriteria proyek yang paling penting dan alternatif yang paling tepat untuk mengatasinya. Kartu skor dibuat dengan menambahkan kolom berlabel “bobot” yang menyertakan angka yang menggambarkan seberapa penting setiap kriteria bagi keputusan akhir. Biasanya, analis mengambil 100 poin dan menyebarkannya ke seluruh kriteria dengan tepat. Jika lima kriteria digunakan dan semuanya sama pentingnya, maka masing-masing kriteria akan mendapat bobot 20. Namun, jika biaya adalah kriteria yang paling penting untuk memilih alternatif, mungkin menerima 60 poin, dan empat kriteria lainnya mungkin hanya mendapat 10 poin.

Kemudian, analis menambahkan kolom yang disebut “Skor” ke dalam matriks yang mengomunikasikan seberapa baik setiap alternatif memenuhi kriteria. Biasanya, rentang angka seperti 1 hingga 5 atau 1 hingga 10 digunakan untuk menilai kelayakan alternatif berdasarkan kriteria. Jadi, untuk kriteria biaya, alternatif yang paling murah dapat menerima 5 pada skala 1-ke-5, sedangkan alternatif yang mahal akan menerima 1. Skor tertimbang dihitung dengan bobot masing-masing kriteria dikalikan dengan skor yang diberikan untuk setiap alternatif. Kemudian, skor tertimbang dijumlahkan untuk setiap alternatif. Skor tertimbang tertinggi mencapai kecocokan terbaik untuk kriteria kami. Ketika angka digunakan dalam matriks alternatif, tim proyek dapat membuat keputusan secara kuantitatif dan berdasarkan angka yang sulit.

Akan tetapi, harus ditunjukkan bahwa skor yang diberikan pada kriteria untuk setiap alternatif tidak lebih dari penilaian subjektif. Akibatnya, sangat mungkin bagi seorang analis untuk memiringkan analisis menurut biasnya sendiri. Dengan kata lain, matriks alternatif berbobot dapat dibuat untuk mendukung alternatif mana pun yang Anda sukai dan tetap mempertahankan tampilan analisis yang objektif dan rasional. Untuk menghindari masalah analisis yang bias, setiap analis dalam tim dapat mengembangkan peringkat secara independen;

kemudian, peringkat dapat dibandingkan dan perbedaan diselesaikan dalam diskusi tim terbuka.

Langkah terakhir, tentu saja, adalah memutuskan solusi mana yang akan dirancang dan diimplementasikan. Keputusan harus dibuat oleh kombinasi pengguna bisnis dan profesional teknis setelah masalah yang terlibat dengan alternatif yang berbeda dipahami dengan baik. Setelah keputusan diselesaikan, desain dapat dilanjutkan sesuai kebutuhan, berdasarkan alternatif yang dipilih.

Evaluation Criteria	Relative Importance (Weight)	Alternative 1: Custom Application Using VB.NET	Score (1–5)*	Weighted Score	Alternative 2: Custom Application Using Java	Score (1–5)*	Weighted Score	Alternative 3: Packaged Software Product ABC	Score (1–5)*	Weighted Score
Technical Issues:										
Criterion 1	20		5	100		3	60		3	60
Criterion 2	10		3	30		3	30		5	50
Criterion 3	10		2	20		1	10		3	30
Economic Issues:										
Criterion 4	25	Supporting	3	75	Supporting	3	75	Supporting	5	125
Criterion 5	10	Information	3	30	Information	1	10	Information	5	50
Organizational Issues:										
Criterion 6	10		5	50		5	50		3	30
Criterion 7	10		3	30		3	30		1	10
Criterion 8	5		3	15		1	5		1	5
TOTAL	100			350			270			360

* This denotes how well the alternative meets the criteria. 1 = poor fit; 5 = perfect fit.

Gambar 7-25 Contoh Matriks Alternatif Menggunakan Bobot

Pertanyaan

1. Jelaskan perbedaan utama antara model analisis dan model desain.
2. Apa yang dimaksud dengan penyeimbangan model?
3. Apa keterkaitan antara model fungsional, struktural, dan perilaku yang perlu diuji?
4. Apa yang dimaksud dengan pemfaktoran? Bagaimana kaitannya dengan abstraksi dan penyempurnaan?
5. Apa itu partisi? Bagaimana partisi berhubungan dengan kolaborasi?
6. Apa itu lapisan? Beri nama lapisan yang berbeda.
7. Apa tujuan dari lapisan yang berbeda?
8. Jelaskan berbagai jenis kelas yang dapat muncul pada setiap lapisan.
9. Masalah atau pertanyaan apa yang muncul pada setiap lapisan yang berbeda?
10. Apa itu paket? Bagaimana paket terkait dengan partisi dan lapisan?
11. Apa itu hubungan ketergantungan? Bagaimana Anda mengidentifikasi mereka?
12. Apa lima langkah untuk mengidentifikasi paket dan membuat diagram paket?
13. Apa yang perlu diverifikasi dan divalidasi dalam diagram paket?
14. Saat menggambar diagram paket, pedoman apa yang harus Anda ikuti?
15. Situasi apa yang paling tepat untuk strategi desain pengembangan kustom?
16. Apa saja masalah dengan menggunakan pendekatan paket perangkat lunak untuk membangun sistem baru? Bagaimana masalah-masalah ini dapat diatasi?
17. Mengapa perusahaan berinvestasi dalam sistem ERP?
18. Apa pro dan kontra menggunakan solusi?
19. Kapan outsourcing dianggap sebagai strategi desain yang baik? Kapan tidak sesuai?
20. Apa itu pembungkus objek?
21. Apa itu integrasi sistem? Jelaskan tantangannya.

22. Apa perbedaan antara kontrak pengaturan waktu, harga tetap, dan nilai tambah untuk outsourcing?
23. Bagaimana matriks alternatif dan analisis kelayakan terkait?
24. Apa itu RFP? Apa bedanya dengan RFI?

Latihan

A. Untuk masalah A Real Estate Inc. di Bab 4 (latihan I, J, dan K), 5 (latihan P dan Q), dan 6 (latihan D):
   1. Lakukan langkah-langkah verifikasi dan validasi model fungsional, struktural, dan perilaku untuk memastikan bahwa semua masalah antar-model telah diselesaikan.
   2. Menggunakan diagram komunikasi dan matriks CRUDE, buat diagram paket dari lapisan domain masalah.
   3. Lakukan langkah-langkah verifikasi dan validasi diagram paket.
   4. Berdasarkan model analisis yang telah dibuat dan pemahaman Anda saat ini tentang posisi perusahaan, strategi desain apa yang akan Anda rekomendasikan? Mengapa?

B. Untuk masalah A Video Store di Bab 4 (latihan L, M, dan N), 5 (latihan R dan S), dan 6 (latihan E):
   1. Lakukan langkah-langkah verifikasi dan validasi model fungsional, struktural, dan perilaku untuk memastikan bahwa semua masalah antar-model telah diselesaikan.
   2. Menggunakan diagram komunikasi dan matriks CRUDE, buat diagram paket dari lapisan domain masalah.
   3. Lakukan langkah-langkah verifikasi dan validasi diagram paket.
   4. Berdasarkan model analisis yang telah dibuat dan pemahaman Anda saat ini tentang posisi perusahaan, strategi desain apa yang akan Anda rekomendasikan? Mengapa?

C. Untuk masalah keanggotaan klub kesehatan di Bab 4 (latihan O, P, dan Q), 5 (latihan T dan U), dan 6 (latihan F):
   1. Lakukan langkah-langkah verifikasi dan validasi model fungsional, struktural, dan perilaku untuk memastikan bahwa semua masalah antar-model telah diselesaikan.
   2. Dengan menggunakan diagram komunikasi dan matriks CRUDE, buat diagram paket dari lapisan domain masalah.
   3. Lakukan langkah-langkah verifikasi dan validasi diagram paket.
   4. Berdasarkan model analisis yang telah dibuat dan pemahaman Anda saat ini tentang posisi perusahaan, strategi desain apa yang akan Anda rekomendasikan? Mengapa?

D. Untuk masalah Piknik R Us di Bab 4 (latihan R, S, dan T), 5 (latihan V dan W), dan 6 (latihan G):
   1. Lakukan langkah-langkah verifikasi dan validasi model fungsional, struktural, dan perilaku untuk memastikan bahwa semua masalah antar-model telah diselesaikan.
   2. Menggunakan diagram komunikasi dan matriks CRUDE, buat diagram paket dari lapisan domain masalah.
   3. Lakukan langkah-langkah verifikasi dan validasi diagram paket.
   4. Berdasarkan model analisis yang telah dibuat dan pemahaman Anda saat ini tentang posisi perusahaan, strategi desain apa yang akan Anda rekomendasikan? Mengapa?

E. Untuk masalah Klub Bulanan di Bab 4 (latihan U, V, dan W), 5 (latihan X dan Y), dan 6 (latihan H):
   1. Lakukan langkah-langkah verifikasi dan validasi model fungsional, struktural, dan perilaku untuk memastikan bahwa semua masalah antar-model telah diselesaikan.

2. Menggunakan diagram komunikasi dan matriks CRUDE, buat diagram paket dari lapisan domain masalah.
3. Lakukan langkah-langkah verifikasi dan validasi diagram paket.
4. Berdasarkan model analisis yang telah dibuat dan pemahaman Anda saat ini tentang posisi perusahaan, strategi desain apa yang akan Anda rekomendasikan? Mengapa?

F. Misalkan Anda memimpin sebuah proyek yang akan menerapkan sistem pendaftaran kursus baru untuk universitas Anda. Anda sedang mempertimbangkan untuk menggunakan aplikasi pendaftaran kursus yang dikemas atau mengalihdayakan pekerjaan ke konsultan eksternal. Buat garis besar untuk RFP yang dapat ditanggapi oleh vendor dan konsultan yang tertarik.

G. Misalkan Anda dan teman Anda memulai bisnis kecil mengecat rumah di musim panas. Anda perlu membeli paket perangkat lunak yang menangani transaksi keuangan bisnis. Buat matriks alternatif yang membandingkan tiga sistem paket (mis., Quicken, MS Money, Quickbooks). Alternatif mana yang tampaknya menjadi pilihan terbaik?

7.7 MINICASE

1. Susan, presiden MOTO, Inc., sebuah perusahaan manajemen sumber daya manusia, merenungkan sistem perangkat lunak manajemen klien yang dibeli organisasinya empat tahun lalu. Pada saat itu, perusahaan baru saja mengalami lonjakan pertumbuhan besar, dan campuran prosedur otomatis dan manual yang telah digunakan untuk mengelola akun klien menjadi berat. Susan dan Nancy, kepala departemen SI-nya, meneliti dan memilih paket yang saat ini digunakan. Susan telah mendengar tentang perangkat lunak pada konferensi profesional yang dia hadiri, dan, setidaknya pada awalnya, itu bekerja cukup baik untuk perusahaan. Beberapa prosedur mereka harus diubah agar sesuai dengan paket, tetapi mereka mengharapkan itu dan siap untuk itu.

   Sejak saat itu, MOTO, Inc., terus berkembang, tidak hanya melalui perluasan basis klien tetapi juga melalui akuisisi beberapa bisnis kecil yang terkait dengan pekerjaan. MOTO, Inc., adalah bisnis yang jauh berbeda dari empat tahun lalu. Seiring dengan perluasan untuk menawarkan layanan manajemen sumber daya manusia yang lebih beragam, staf pendukung perusahaan juga telah berkembang. Susan dan Nancy sangat bangga dengan departemen IS yang telah mereka bangun selama bertahun-tahun. Menggunakan ikatan yang kuat dengan universitas lokal, paket kompensasi yang menarik, dan lingkungan kerja yang baik, departemen IS memiliki staf yang baik dengan orang-orang yang kompeten dan inovatif, ditambah aliran magang perguruan tinggi yang membuat departemen tetap segar dan hidup. Salah satu tim IS memelopori penggunaan Internet untuk menawarkan layanan MOTO ke segmen pasar yang sama sekali baru, sebuah eksperimen yang terbukti sangat sukses.

   Tampaknya jelas bahwa perubahan besar diperlukan dalam perangkat lunak manajemen klien, dan Susan telah mulai merencanakan keuangan untuk melakukan proyek semacam itu. Perangkat lunak ini adalah bagian utama dari operasi MOTO, dan Susan ingin memastikan bahwa sistem berkualitas tinggi diperoleh kali ini. Dia tahu bahwa vendor sistem mereka saat ini telah membuat beberapa revisi dan penambahan pada lini produknya. Sejumlah vendor perangkat lunak lain juga menawarkan produk yang mungkin cocok. Beberapa vendor tersebut tidak ada saat pembelian dilakukan empat tahun lalu. Susan juga mempertimbangkan saran Nancy agar departemen SI mengembangkan aplikasi perangkat lunak khusus.

a. Garis besar isu-isu yang Susan harus pertimbangkan yang akan mendukung pengembangan aplikasi perangkat lunak kustom in-house.
b. Uraikan masalah yang harus dipertimbangkan Susan yang akan mendukung pembelian paket perangkat lunak.
c. Dalam konteks proyek pengembangan sistem, kapan keputusan make-versus-buy harus dibuat? Bagaimana Susan melanjutkan? Jelaskan jawabanmu.

2. Lihat minicase 1 (West Star Marinas) di Bab 5. Setelah semua model analisis (baik model as-is dan to-be) untuk West Star Marinas selesai, direktur operasi akhirnya mengerti mengapa penting untuk memahami sistem saat in sebelum mempelajari pengembangan sistem yang akan datang. Namun, Anda sekarang memberi tahu dia bahwa calon model hanyalah bagian domain masalah dari desain. Dia sekarang sangat bingung. Setelah menjelaskan kepadanya keuntungan menggunakan pendekatan berlapis untuk mengembangkan sistem, dia berkata, "Saya tidak peduli tentang penggunaan kembali atau pemeliharaan. Saya hanya ingin sistem ini diterapkan sesegera mungkin. Anda tipe IS selalu mencoba menarik yang cepat pada pengguna. Selesaikan saja sistemnya."

   Apa tanggapan Anda terhadap Direktur Operasi? Apakah Anda terjun ke implementasi seperti yang dia inginkan? Apa yang Anda lakukan selanjutnya?

3. Lihat model analisis yang Anda buat untuk manajemen staf profesional dan ilmiah (PSSM) untuk kasus kecil 2 di Bab 4 dan untuk kasus kecil 1 di Bab 6.
   a. Lakukan langkah-langkah verifikasi dan validasi model fungsional, struktural, dan perilaku untuk memastikan bahwa semua masalah antar-model telah diselesaikan.
   b. Menggunakan diagram komunikasi dan matriks CRUDE, buat diagram paket dari lapisan domain masalah.
   c. Lakukan langkah-langkah verifikasi dan validasi diagram paket.
   d. Berdasarkan model analisis yang telah dibuat dan pemahaman Anda saat ini tentang posisi perusahaan, strategi desain apa yang akan Anda rekomendasikan? Mengapa?
4. Lihat model analisis yang Anda buat untuk Kendaraan Perjalanan Liburan untuk minicase 2 di Bab 5 dan untuk minicase 2 di Bab 6.
   a. Lakukan langkah-langkah verifikasi dan validasi model fungsional, struktural, dan perilaku untuk memastikan bahwa semua masalah antar-model telah diselesaikan.
   b. Menggunakan diagram komunikasi dan matriks CRUDE, buat diagram paket dari lapisan domain masalah.
   c. Lakukan langkah-langkah verifikasi dan validasi diagram paket.
   d. Berdasarkan model analisis yang telah dibuat dan pemahaman Anda saat ini tentang posisi perusahaan, strategi desain apa yang akan Anda rekomendasikan? Mengapa?

BAB 8

DESAIN KELAS DAN METODE

Langkah terpenting dari fase desain adalah mendesain kelas dan metode individual. Sistem berorientasi objek bisa sangat kompleks, sehingga analis perlu membuat instruksi dan pedoman untuk pemrogram yang dengan jelas menggambarkan apa yang harus dilakukan sistem. Bab ini menyajikan seperangkat kriteria, aktivitas, dan teknik yang digunakan untuk merancang kelas dan metode. Bersama-sama mereka digunakan untuk memastikan bahwa desain berorientasi objek mengkomunikasikan bagaimana sistem perlu dikodekan.

8.1 Tujuan

- Menjadi akrab dengan istilah coupling, kohesi, dan connascence.
- Mampu menentukan, merestrukturisasi, dan mengoptimalkan desain objek.
- Mampu mengidentifikasi penggunaan kembali kelas, perpustakaan, kerangka kerja, dan komponen yang telah ditentukan sebelumnya.
- Mampu menentukan batasan dan kontrak.
- Mampu membuat spesifikasi metode.

8.2 Pendahuluan

Saat ini, banyak implementasi aktual akan dilakukan di lokasi geografis yang berbeda dari tempat analisis dan desain dilakukan. Kita harus memastikan bahwa desain ditentukan dengan cara yang “benar” dan tidak ada, atau setidaknya minimal, ambiguitas dalam spesifikasi desain.

Di dunia yang datar saat ini, bahasa umum yang digunakan di antara pengembang kemungkinan besar adalah UML dan beberapa bahasa berorientasi objek, seperti Java, dan bukan bahasa Inggris. Bahasa Inggris selalu dan akan selalu ambigu. Selanjutnya, untuk variasi bahasa Inggris apa yang kita rujuk? Seperti yang ditunjukkan oleh Oscar Wilde dan George Bernard Shaw secara independen, Amerika Serikat dan Inggris dibagi oleh bahasa yang sama.

Secara praktis, desain Kelas dan Metode adalah tempat semua pekerjaan benar-benar diselesaikan selama desain. Tidak peduli lapisan mana yang Anda fokuskan, kelas, yang akan digunakan untuk membuat objek sistem, harus dirancang. Beberapa orang percaya bahwa dengan perpustakaan kelas yang dapat digunakan kembali dan komponen siap pakai, jenis desain tingkat rendah, atau terperinci, ini membuang-buang waktu dan bahwa kita harus segera terjun ke pekerjaan "nyata": pengkodean sistem. Namun, pengalaman masa lalu menunjukkan bahwa desain tingkat rendah, atau detail, sangat penting meskipun menggunakan pustaka dan komponen. Desain detail masih sangat penting karena tiga alasan. Pertama, dengan CASE tool modern saat ini, cukup banyak kode aktual yang dapat dihasilkan oleh alat dari desain detail. Kedua, bahkan kelas dan komponen yang sudah ada sebelumnya perlu dipahami, diatur, dan disatukan. Ketiga, masih umum bagi tim proyek untuk menulis beberapa kode dan menghasilkan kelas asli yang mendukung logika aplikasi sistem.

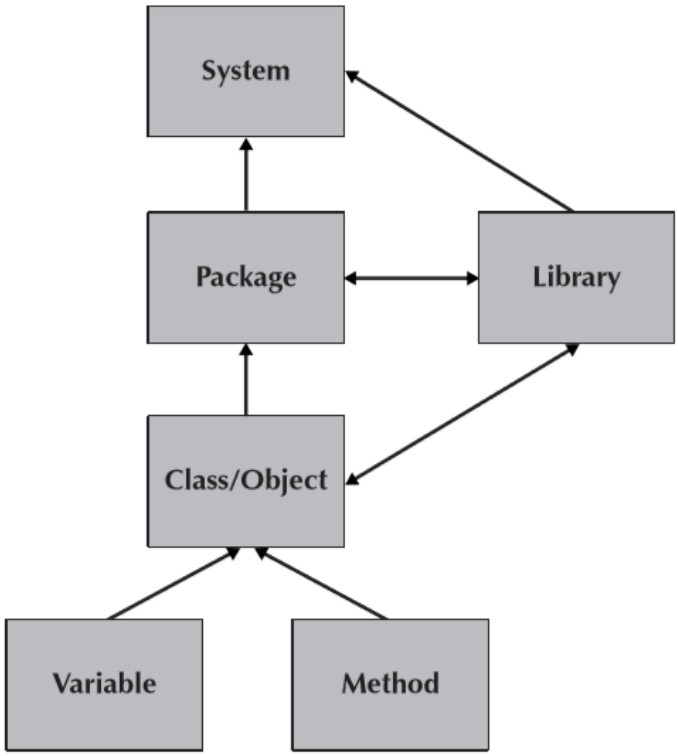
Melompat langsung ke pengkodean akan menjamin hasil yang buruk. Misalnya, meskipun penggunaan lapisan dapat menyederhanakan kelas individu, mereka dapat meningkatkan kompleksitas interaksi di antara mereka. Jika kelas tidak dirancang dengan hati-hati, sistem yang dihasilkan bisa sangat tidak efisien. Atau lebih buruk lagi, instance kelas

(yaitu, objek) tidak akan mampu berkomunikasi satu sama lain, yang akan mengakibatkan sistem tidak berfungsi dengan baik.

Dalam sistem berorientasi objek, perubahan dapat terjadi pada tingkat abstraksi yang berbeda. Level-level ini termasuk variabel, metode, kelas/objek, paket,1 perpustakaan, dan/atau level aplikasi/sistem (lihat Gambar 8-1). Perubahan yang terjadi pada satu tingkat dapat mempengaruhi tingkat lain (misalnya, perubahan kelas dapat mempengaruhi tingkat paket, yang dapat mempengaruhi tingkat sistem dan tingkat perpustakaan, yang pada gilirannya dapat menyebabkan perubahan kembali di tingkat kelas.). Akhirnya, perubahan dapat terjadi pada tingkat yang berbeda pada waktu yang sama.

Kabar baiknya adalah bahwa desain rinci dari masing-masing kelas dan metode cukup mudah. Interaksi antara objek pada lapisan domain masalah telah dirancang, dalam beberapa detail, selama analisis (lihat Bab 4 sampai 6). Lapisan lain (manajemen data, interaksi manusia-komputer, dan arsitektur fisik) sangat bergantung pada lapisan domain masalah. Oleh karena itu, jika kelas domain masalah dirancang dengan benar, desain kelas pada lapisan lain akan sesuai, secara relatif.

Karena itu, menurut pengalaman kami, banyak tim proyek terlalu cepat dalam menulis kode untuk kelas tanpa mendesainnya terlebih dahulu. Beberapa di antaranya disebabkan oleh fakta bahwa analisis dan desain sistem berorientasi objek telah berevolusi dari pemrograman berorientasi objek. Sampai saat ini ada kekurangan pedoman umum tentang bagaimana merancang dan mengembangkan sistem berorientasi objek yang efektif. Namun, dengan penerimaan UML sebagai notasi objek standar, pendekatan standar berdasarkan karya banyak metodologi objek mulai muncul.



Gambar 8-1 Tingkatan Abstraksi dalam Sistem Berorientasi Objek

Sumber: Berdasarkan materi dari David P. Tegarden, Steven D. Sheetz, dan David E. Monarchi, “A Software Complexity Model of Object-Oriented Systems,” Sistem Pendukung Keputusan 13 (Maret 1995): 241–262.

8.3 TINJAUAN KARAKTERISTIK DASAR ORIENTASI OBYEK

Sistem berorientasi objek dapat ditelusuri kembali ke bahasa pemrograman Simula dan Smalltalk. Namun, sampai peningkatan daya prosesor dan penurunan biaya prosesor yang terjadi pada 1980-an, pendekatan berorientasi objek tidak praktis. Banyak detail spesifik mengenai karakteristik dasar orientasi objek bergantung pada bahasa; yaitu, setiap bahasa pemrograman berorientasi objek cenderung mengimplementasikan beberapa dasar berorientasi objek dengan cara yang berbeda. Akibatnya, kita perlu mengetahui bahasa pemrograman mana yang akan digunakan untuk mengimplementasikan berbagai aspek solusi. Jika tidak, sistem dapat berperilaku dengan cara yang berbeda dari yang diharapkan oleh analis, perancang, dan klien. Saat ini, bahasa pemrograman C++, Java, Objective-C, dan Visual Basic cenderung menjadi bahasa yang lebih dominan digunakan. Di bagian ini, kami meninjau karakteristik dasar orientasi objek dan menunjukkan di mana masalah khusus bahasa muncul.

Kelas, Objek, Metode, dan Pesan

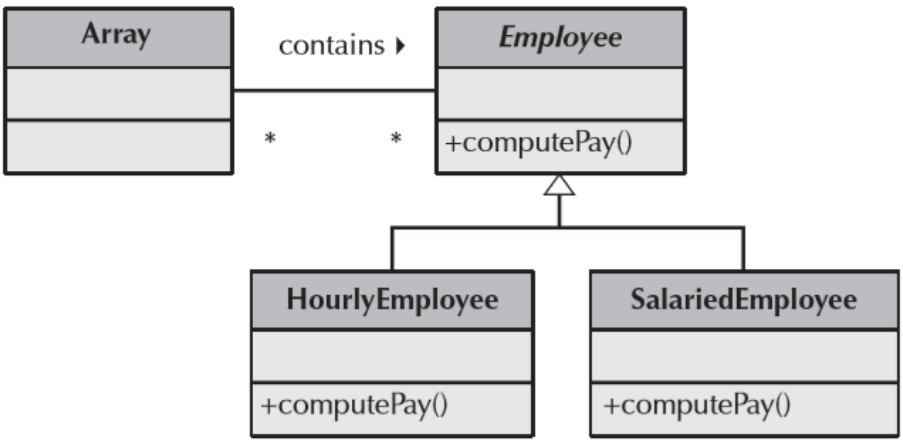
Blok bangunan dasar dari sistem adalah objek. Objek adalah turunan dari kelas. Kelas adalah template yang kita gunakan untuk mendefinisikan data dan proses yang dikandung setiap objek. Setiap objek memiliki atribut yang menggambarkan data tentang objek tersebut. Objek memiliki status, yang ditentukan oleh nilai atributnya dan hubungannya dengan objek lain pada titik waktu tertentu. Dan setiap objek memiliki metode, yang menentukan proses apa yang dapat dilakukan objek. Dari perspektif kami, metode digunakan untuk mengimplementasikan operasi yang menentukan perilaku objek (lihat Bab 5). Untuk mendapatkan objek untuk melakukan metode (misalnya, untuk menghapus sendiri), pesan dikirim ke objek. Pesan pada dasarnya adalah panggilan fungsi atau prosedur dari satu objek ke objek lain.

Enkapsulasi dan Penyembunyian Informasi

Enkapsulasi adalah mekanisme yang menggabungkan proses dan data menjadi satu objek. Penyembunyian informasi menunjukkan bahwa hanya informasi yang diperlukan untuk menggunakan suatu objek yang tersedia di luar objek; yaitu, penyembunyian informasi terkait dengan visibilitas metode dan atribut (lihat Bab 5). Bagaimana tepatnya objek menyimpan data atau melakukan metode tidak relevan, selama objek berfungsi dengan benar. Semua yang diperlukan untuk menggunakan objek adalah kumpulan metode dan pesan yang perlu dikirim untuk memicunya. Satu-satunya komunikasi antara objek harus melalui metode objek. Fakta bahwa kita dapat menggunakan suatu objek dengan mengirimkan pesan yang memanggil metode adalah utama untuk dapat digunakan kembali karena melindungi kerja internal objek dari perubahan di sistem luar, dan menjaga sistem agar tidak terpengaruh ketika perubahan dibuat ke obyek.

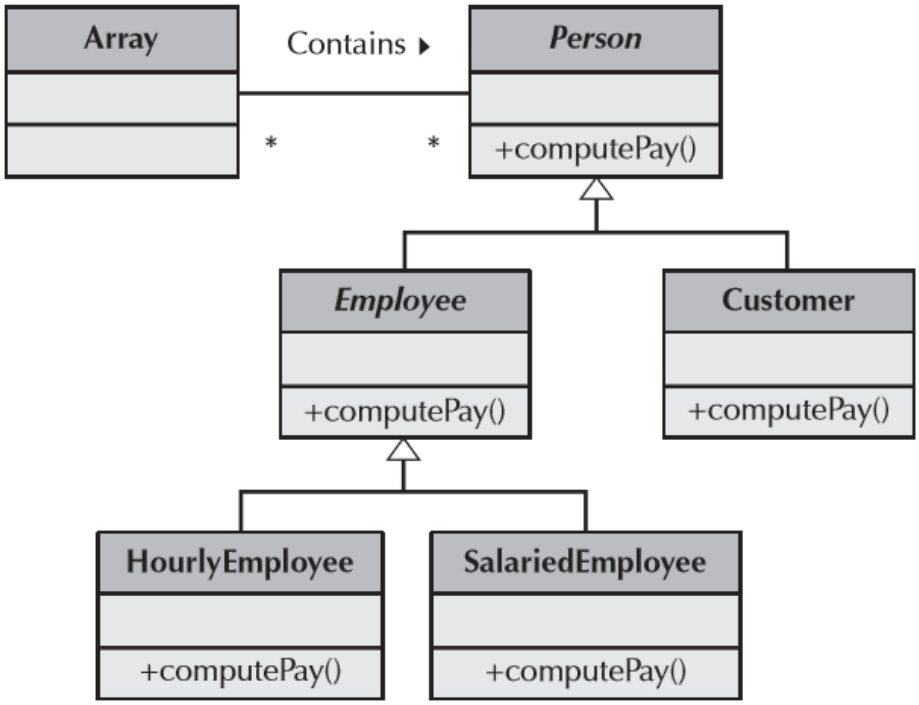
Polimorfisme dan Pengikatan Dinamis

Polimorfisme berarti memiliki kemampuan untuk mengambil beberapa bentuk. Dengan mendukung polimorfisme, sistem berorientasi objek dapat mengirim pesan yang sama ke sekumpulan objek, yang dapat diinterpretasikan secara berbeda oleh kelas objek yang berbeda. Berdasarkan enkapsulasi dan penyembunyian informasi, suatu objek tidak harus peduli dengan bagaimana sesuatu dilakukan saat menggunakan objek lain. Itu hanya mengirim pesan ke suatu objek dan objek itu menentukan bagaimana menafsirkan pesan. Ini dicapai melalui penggunaan pengikatan dinamis.



Gambar 8-2 Contoh Polimorfisme

Pengikatan dinamis mengacu pada kemampuan sistem berorientasi objek untuk menunda pengetikan data objek untuk menjalankan waktu. Misalnya, bayangkan Anda memiliki sebuah array tipe karyawan yang berisi contoh karyawan per jam dan karyawan yang digaji (lihat Gambar 8-2). Kedua jenis karyawan ini menerapkan metode pembayaran komputasi. Sebuah objek dapat mengirim pesan ke setiap instance yang terdapat dalam array untuk menghitung pembayaran untuk masing-masing instance tersebut. Bergantung pada apakah instans adalah karyawan per jam atau karyawan bergaji, metode yang berbeda akan dijalankan. Metode spesifik dipilih pada saat run time. Dengan kemampuan ini, kelas individu lebih mudah dipahami. Namun, tingkat dukungan khusus untuk polimorfisme dan pengikatan dinamis adalah khusus bahasa. Sebagian besar bahasa pemrograman berorientasi objek mendukung metode pengikatan dinamis, dan beberapa mendukung pengikatan atribut secara dinamis.



Gambar 8-3 Contoh Penyalahgunaan Polimorfisme

Tapi polimorfisme bisa menjadi pedang bermata dua. Melalui penggunaan dynamic binding, tidak ada cara untuk mengetahui sebelum run time objek spesifik mana yang akan diminta untuk mengeksekusi metodenya. Akibatnya, ada keputusan yang dibuat oleh sistem yang tidak dikodekan di mana pun. Karena semua keputusan ini dibuat pada saat run time, dimungkinkan untuk mengirim pesan ke objek yang tidak dipahaminya (yaitu, objek tidak memiliki metode yang sesuai). Ini dapat menyebabkan kesalahan run-time yang, jika sistem tidak diprogram untuk menanganinya dengan benar, dapat menyebabkan sistem dibatalkan.

Terakhir, jika metode tidak konsisten secara semantik, pengembang tidak dapat berasumsi bahwa semua metode dengan nama yang sama akan melakukan operasi generik yang sama. Misalnya, bayangkan Anda memiliki array tipe orang yang berisi contoh karyawan dan pelanggan (lihat Gambar 8-3). Keduanya menerapkan metode pembayaran komputasi. Sebuah objek dapat mengirim pesan ke setiap instans yang terdapat dalam larik untuk mengeksekusi metode pembayaran komputasi untuk instans individual tersebut. Dalam kasus karyawan, metode pembayaran komputasi menghitung jumlah hutang karyawan oleh perusahaan, sedangkan metode pembayaran komputasi yang terkait dengan instans pelanggan menghitung jumlah hutang perusahaan oleh pelanggan. Bergantung pada apakah instance adalah karyawan atau pelanggan, arti yang berbeda dikaitkan dengan metode tersebut. Oleh karena itu, semantik setiap metode harus ditentukan secara individual. Hal ini secara substansial meningkatkan kesulitan memahami objek individu. Utama untuk mengendalikan kesulitan memahami sistem berorientasi objek saat menggunakan polimorfisme adalah memastikan bahwa semua metode dengan nama yang sama mengimplementasikan operasi generik yang sama (yaitu, secara semantik konsisten).

Warisan

Warisan memungkinkan pengembang untuk mendefinisikan kelas secara bertahap dengan menggunakan kembali kelas yang didefinisikan sebelumnya sebagai dasar untuk kelas baru. Meskipun kita dapat mendefinisikan setiap kelas secara terpisah, mungkin lebih mudah untuk mendefinisikan satu superclass umum yang berisi data dan metode yang dibutuhkan oleh subclass dan kemudian membuat kelas-kelas ini mewarisi properti superclass. Subclass mewarisi atribut dan metode dari superclass di atasnya. Warisan membuatnya lebih mudah untuk mendefinisikan kelas.

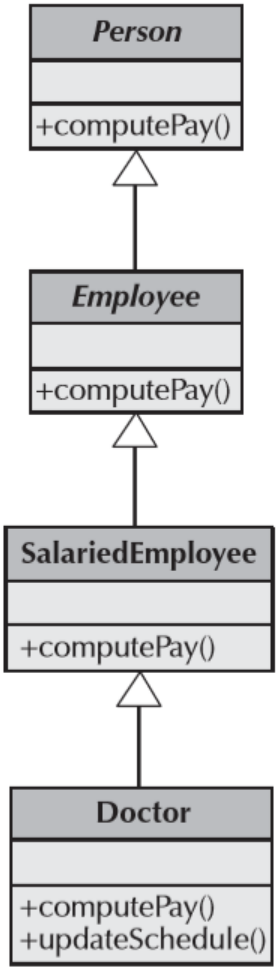
Ada banyak jenis mekanisme pewarisan yang terkait dengan sistem berorientasi objek. Mekanisme pewarisan yang paling umum mencakup berbagai bentuk pewarisan tunggal dan ganda. Warisan tunggal memungkinkan subkelas hanya memiliki satu kelas induk. Saat ini, semua metodologi berorientasi objek, database, dan bahasa pemrograman mengizinkan perluasan definisi superclass melalui pewarisan tunggal.

Beberapa metodologi berorientasi objek, database, dan bahasa pemrograman memungkinkan subkelas untuk mendefinisikan kembali beberapa atau semua atribut dan/atau metode superkelasnya. Dengan kemampuan redefinisi, dimungkinkan untuk memperkenalkan konflik pewarisan [yaitu, atribut (atau metode) dari subkelas dengan nama yang sama dengan atribut (atau metode) dari kelas super]. Sebagai contoh pada Gambar 8-4, Doctor adalah subclass dari Employee. Keduanya memiliki metode bernama ComputePay(). Hal ini menyebabkan konflik pewarisan. Selanjutnya, ketika definisi superclass dimodifikasi, semua subclass akan terpengaruh. Hal ini dapat menimbulkan konflik pewarisan tambahan dalam satu (atau lebih) subkelas superclass. Misalnya pada Gambar 8-4, Karyawan dapat dimodifikasi untuk menyertakan metode tambahan, UpdateSchedule(). Ini akan menambah konflik warisan lain antara Karyawan dan Dokter. Oleh karena itu, pengembang harus mewaspadai efek modifikasi tidak hanya pada superclass tetapi juga pada setiap subclass yang mewarisi modifikasi tersebut.

Akhirnya, melalui kemampuan redefinisi, pemrogram dapat membatalkan pewarisan metode secara sewenang-wenang dengan menempatkan stub di subkelas yang akan menggantikan definisi metode yang diwarisi. Jika pembatalan metode diperlukan untuk definisi subclass yang benar, maka kemungkinan subclass telah salah diklasifikasikan (yaitu, mewarisi dari superclass yang salah).

Seperti yang Anda lihat, dari perspektif desain, konflik pewarisan dan pendefinisian ulang dapat menyebabkan semua jenis masalah dalam menafsirkan desain dan implementasi akhir.7 Namun, sebagian besar konflik pewarisan disebabkan oleh klasifikasi subkelas yang buruk dalam hierarki pewarisan (generalisasi a -jenis semantik dilanggar), atau mekanisme pewarisan yang sebenarnya melanggar prinsip enkapsulasi dan penyembunyian informasi (yaitu, subkelas mampu menangani atribut atau metode superkelas secara langsung). Untuk mengatasi masalah ini, Jim Rumbaugh dan rekan-rekannya menyarankan pedoman berikut:

- Jangan mendefinisikan ulang operasi kueri.
- Metode yang mendefinisikan ulang yang diwariskan harus membatasi hanya semantik dari yang diwariskan.
- Semantik yang mendasari metode yang diwarisi tidak boleh diubah.
- Tanda tangan (daftar argumen) dari metode yang diwarisi tidak boleh diubah.



Gambar 8-4 Contoh Redefinisi dan Konflik Pewarisan

Namun, banyak bahasa pemrograman berorientasi objek yang ada melanggar pedoman ini. Ketika datang untuk mengimplementasikan desain, bahasa pemrograman berorientasi

objek yang berbeda menangani konflik pewarisan secara berbeda. Oleh karena itu, penting pada titik ini dalam pengembangan sistem untuk mengetahui apa yang didukung oleh bahasa pemrograman yang dipilih. Kita harus yakin bahwa desain dapat diimplementasikan sebagaimana dimaksud. Jika tidak, desain perlu dimodifikasi sebelum diserahkan ke pemrogram yang berlokasi jauh.

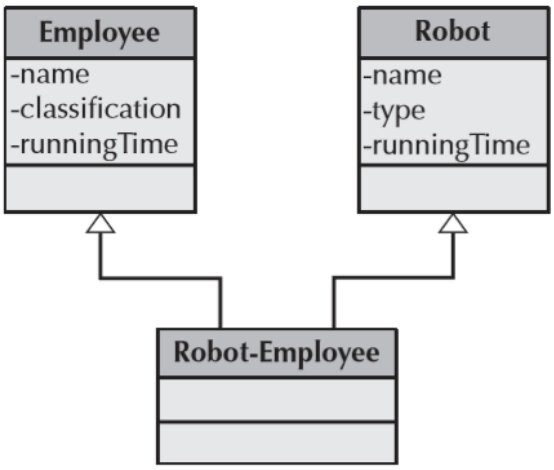
Ketika mempertimbangkan interaksi pewarisan dengan polimorfisme dan pengikatan dinamis, sistem berorientasi objek memberi pengembang seperangkat alat yang sangat kuat, tetapi berbahaya. Tergantung pada bahasa pemrograman berorientasi objek yang digunakan, interaksi ini dapat memungkinkan objek yang sama untuk diasosiasikan dengan kelas yang berbeda pada waktu yang berbeda. Misalnya, instance Doctor dapat diperlakukan sebagai instance Employee atau superclass langsung dan tidak langsungnya, seperti SalriedEmployee dan Person, masing-masing (lihat Gambar 8-4). Oleh karena itu, tergantung pada apakah pengikatan statis atau dinamis didukung, objek yang sama dapat menjalankan implementasi yang berbeda dari metode yang sama pada waktu yang berbeda. Atau, jika metode didefinisikan hanya dengan kelas SalariedEmployee dan saat ini diperlakukan sebagai turunan dari kelas Karyawan, instance dapat menyebabkan kesalahan run-time terjadi. Penting untuk mengetahui bahasa pemrograman berorientasi objek apa yang akan digunakan sehingga masalah seperti ini dapat diselesaikan dengan desain, alih-alih implementasi, kelas.

Dengan multiple inheritance, sebuah subclass dapat mewarisi lebih dari satu superclass. Dalam situasi ini, jenis konflik pewarisan berlipat ganda. Selain kemungkinan memiliki konflik pewarisan antara subkelas dan satu (atau lebih) superkelasnya, sekarang dimungkinkan untuk memiliki konflik antara dua (atau lebih) superkelas. Dalam kasus terakhir ini, tiga jenis konflik pewarisan tambahan dapat terjadi:

Dua atribut (atau metode) yang diwarisi memiliki nama (ejaan) dan semantik yang sama.

Dua atribut (atau metode) yang diwariskan memiliki nama yang berbeda tetapi semantik yang identik (yaitu, keduanya adalah sinonim).

Dua atribut (atau metode) yang diwariskan memiliki nama yang sama tetapi semantik yang berbeda (yaitu, mereka adalah heteronim, homograf, atau homonim). Ini juga melanggar penggunaan polimorfisme yang tepat.



Gambar 8-5 Konflik Warisan Tambahan dengan Banyak Warisan

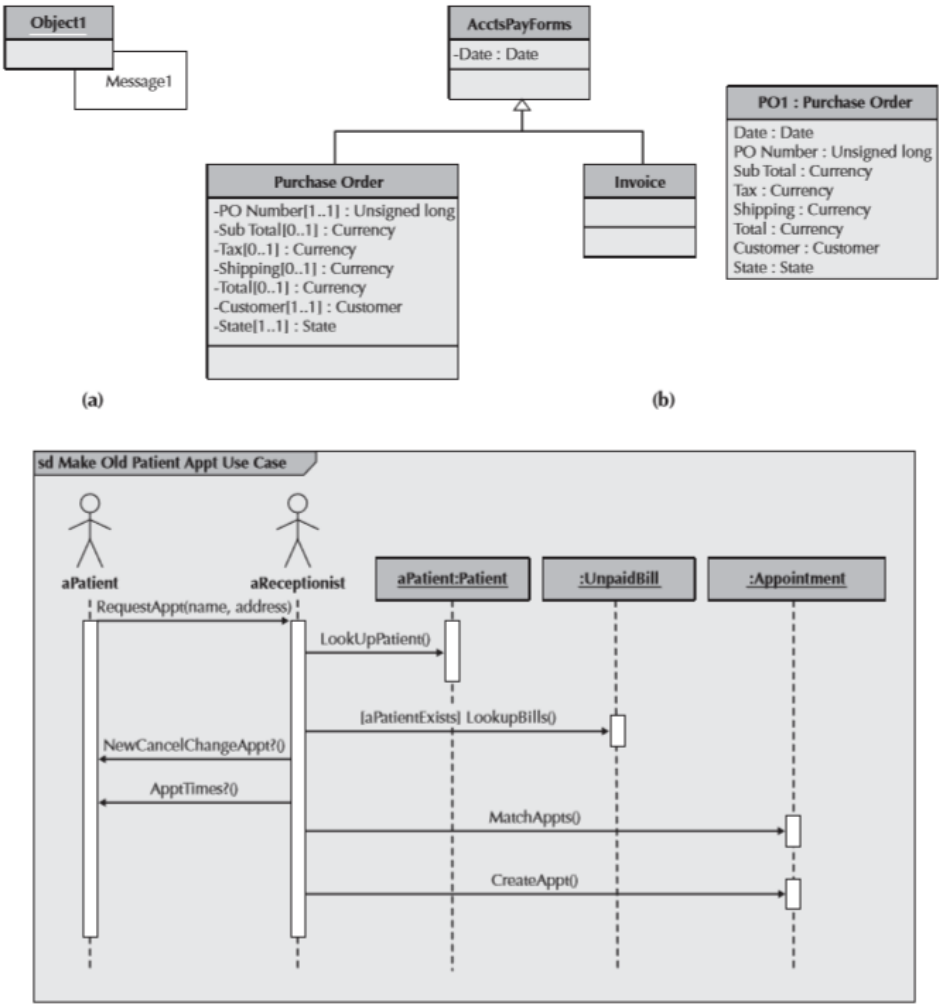
Misalnya, pada Gambar 8-5, Robot-Karyawan adalah subkelas dari Karyawan dan Robot. Dalam hal ini, Karyawan dan Robot bertentangan dengan nama atribut. Yang mana yang harus diwarisi Robot-Karyawan? Karena mereka sama, secara semantik, apakah itu penting? Ada juga kemungkinan bahwa Karyawan dan Robot dapat memiliki konflik semantik pada atribut klasifikasi dan tipe jika mereka memiliki semantik yang sama. Secara praktis, satu-satunya cara untuk mencegah situasi ini adalah pengembang menangkapnya selama desain subkelas. Terakhir, bagaimana jika atribut runningTime memiliki semantik yang berbeda? Dalam kasus objek Karyawan, atribut runningTime menyimpan waktu karyawan yang berlari satu mil, sedangkan atribut runningTime untuk objek Robot menyimpan waktu rata-rata antara pemeriksaan. Haruskah Robot-Karyawan mewarisi keduanya? Itu sangat tergantung pada apakah karyawan robot dapat berlari sejauh satu mil atau tidak. Dengan potensi jenis konflik tambahan ini, ada risiko penurunan pemahaman dalam sistem berorientasi objek alih-alih meningkatkannya melalui penggunaan pewarisan berganda. Saran kami adalah berhati-hatilah saat menggunakan pewarisan berganda.

8.4 KRITERIA DESAIN

Ketika mempertimbangkan desain sistem berorientasi objek, ada seperangkat kriteria yang dapat digunakan untuk menentukan apakah desain itu baik atau buruk. Menurut Coad dan Yourdon, "Desain yang baik adalah desain yang menyeimbangkan trade-off untuk meminimalkan total biaya sistem selama masa pakainya." Kriteria tersebut meliputi coupling, cohesion, dan connascence.

Coupling

Coupling mengacu pada bagaimana saling tergantung atau saling terkait modul (kelas, objek, dan metode) dalam suatu sistem. Semakin tinggi saling ketergantungan, semakin besar kemungkinan perubahan di bagian desain dapat menyebabkan perubahan diperlukan di bagian lain dari desain. Untuk sistem berorientasi objek, Coad dan Yourdon mengidentifikasi dua jenis coupling untuk dipertimbangkan: interaksi dan pewarisan.



Gambar 8-6 Contoh Penggabungan Interaksi

Penggabungan interaksi berkaitan dengan penggabungan antara metode dan objek melalui pengiriman pesan. Lieberherr dan Holland mengajukan hukum Demeter sebagai pedoman untuk meminimalkan jenis coupling ini. Pada dasarnya, hukum meminimalkan jumlah objek yang dapat menerima pesan dari objek tertentu. Hukum menyatakan bahwa suatu objek harus mengirim pesan hanya ke salah satu dari berikut ini:

- Sendiri (Misalnya pada Gambar 8-6a, Object1 dapat mengirim Message1 ke dirinya sendiri. Dengan kata lain, metode yang terkait dengan Object1 dapat menggunakan metode lain yang terkait dengan Object1.)
- Objek yang terkandung dalam atribut objek atau salah satu superclassnya (Misalnya pada Gambar 8-6b, instance P01 dari kelas Purchase Order harus dapat mengirim pesan menggunakan atribut Customer, State, dan Date.)
- Objek yang diteruskan sebagai parameter ke metode (Misalnya pada Gambar 8-6c, instance aPatient mengirim pesan RequestAppt(name, address) ke instance aReceptionist, yang diizinkan untuk mengirim pesan ke instance yang terkandung dalam nama dan parameter alamat.)
- Objek yang dibuat oleh metode (Misalnya pada Gambar 8-6c, metode RequestAppt yang terkait dengan instance aReceptionist membuat instance kelas Appointment. Metode RequestAppt diizinkan untuk mengirim pesan ke instance tersebut.)
- Objek yang disimpan dalam variabel global global

Meskipun hukum Demeter mencoba untuk meminimalkan coupling interaksi antara metode dan objek, masing-masing bentuk pengiriman pesan yang diizinkan di atas sebenarnya meningkatkan coupling. Misalnya, coupling meningkat di antara objek jika metode pemanggilan meneruskan atribut ke metode yang dipanggil atau jika metode pemanggilan bergantung pada nilai yang dikembalikan oleh metode yang dipanggil.

Ada enam jenis coupling interaksi, masing-masing jatuh pada bagian yang berbeda dari kontinum baik-ke-buruk. Mereka berkisar dari tidak ada sambungan langsung hingga sambungan konten. Gambar 8-7 menyajikan berbagai jenis coupling interaksi. Secara umum, interaksi coupling harus diminimalkan. Satu-satunya pengecualian yang mungkin adalah bahwa kelas-kelas domain-non-masalah harus digabungkan ke kelas-kelas domain-masalah yang sesuai. Misalnya, objek laporan (pada lapisan interaksi manusia-komputer) yang menampilkan konten objek karyawan (pada lapisan domain masalah) akan bergantung pada objek karyawan. Dalam hal ini, untuk tujuan pengoptimalan, kelas laporan mungkin berisi konten atau digabungkan secara patologis ke kelas karyawan. Namun, kelas domain masalah tidak boleh digabungkan ke kelas domain non-masalah.

Level	Tipe	Diskripsi
Good	No Direct Coupling	Metode tidak berhubungan satu sama lain; yaitu, mereka tidak saling memanggil.
	Data	Metode pemanggilan meneruskan variabel ke metode yang dipanggil. Jika variabel adalah komposit (yaitu, objek), seluruh objek digunakan oleh metode yang dipanggil untuk menjalankan fungsinya.
	Stamp	Metode pemanggilan melewatkan variabel komposit (yaitu, objek) ke metode yang dipanggil, tetapi metode yang dipanggil hanya menggunakan sebagian dari objek untuk menjalankan fungsinya.
	Control	Metode pemanggilan melewati variabel kontrol yang nilainya akan mengontrol eksekusi metode yang dipanggil.
	Common or Global	Metode mengacu pada "global data area" yang berada di luar objek individu.
Bad	Content or Pathological	Metode dari satu objek mengacu pada bagian dalam (bagian tersembunyi) dari objek lain. Ini melanggar prinsip enkapsulasi dan penyembunyian informasi. Namun, C++ memungkinkan ini terjadi melalui penggunaan "friens"

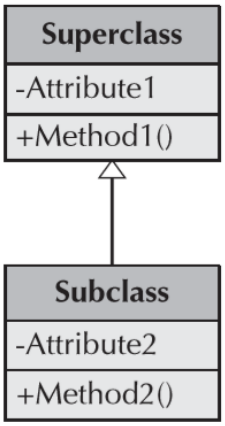
Source: These types are based on material from Meilir Page-Jones, *The Practical Guide to Structured Systems Design,* 2nd Ed. (Englewood Cliffs, NJ: Yardon Press, 1988); Glenford Myers, *Composite/ Structured Design* (New York: Van Nostrand Reinhold, 1978).

Gambar 8-7 Jenis Interaksi Coupling

Inheritance Coupling, seperti namanya, berkaitan dengan seberapa erat kelas-kelas digabungkan dalam hierarki pewarisan. Sebagian besar penulis cenderung mengatakan secara sederhana bahwa jenis coupling ini diinginkan. Namun, tergantung pada masalah yang diangkat sebelumnya dengan pewarisan—konflik pewarisan, kemampuan pendefinisian ulang, dan pengikatan dinamis—penggandaan pewarisan tingkat tinggi mungkin bukan hal yang baik. Sebagai contoh, pada Gambar 8-8, apakah Method2() yang didefinisikan dalam Subclass diperbolehkan untuk memanggil Method1() yang didefinisikan dalam Superclass? Atau, haruskah Method2() didefinisikan di Subclass merujuk ke Attribute1 yang didefinisikan di Superclass? Atau, yang lebih membingungkan, dengan asumsi bahwa Superclass adalah kelas abstrak, dapatkah Method1() memanggil Method2() atau menggunakan Attribute2 yang didefinisikan di Subclass? Jelas, dua contoh pertama memiliki beberapa pengertian intuitif. Menggunakan properti dari superclass adalah tujuan utama dari mewarisi dari itu di tempat pertama. Di sisi lain, contoh ketiga agak berlawanan dengan intuisi. Namun, karena bahasa

pemrograman berorientasi objek yang berbeda mendukung pengikatan dinamis, polimorfisme, dan pewarisan, semua contoh ini dapat dimungkinkan.

Seperti yang telah ditunjukkan Snyder, sebagian besar masalah dengan pewarisan melibatkan kemampuan dalam bahasa pemrograman berorientasi objek untuk melanggar prinsip enkapsulasi dan penyembunyian informasi. Dari perspektif desain, pengembang perlu mengoptimalkan pertukaran dari pelanggaran enkapsulasi dan prinsip penyembunyian informasi dan meningkatkan coupling yang diinginkan antara subclass dan superclass-nya. Cara terbaik untuk memecahkan teka-teki ini adalah memastikan bahwa pewarisan hanya digunakan untuk mendukung semantik generalisasi/spesialisasi (a-kind-of) dan prinsip substitusi (lihat Bab 5). Semua kegunaan lain harus dihindari.



Gambar 8-8 Contoh Inheritance Coupling

Kohesi

Kohesi mengacu pada bagaimana satu modul (kelas, objek, atau metode) berada dalam suatu sistem. Kelas atau objek harus mewakili hanya satu hal, dan metode harus menyelesaikan hanya satu tugas. Tiga tipe umum kohesi telah diidentifikasi oleh Coad dan Yourdon untuk sistem berorientasi objek: metode, kelas, dan generalisasi/spesialisasi.

Kohesi metode membahas kohesi dalam metode individual (yaitu, seberapa berpikiran tunggal suatu metode). Metode harus melakukan satu dan hanya satu hal. Metode yang benar-benar melakukan banyak fungsi lebih sulit untuk dipahami—dan, oleh karena itu, untuk diterapkan dan dipelihara—daripada metode yang hanya melakukan satu fungsi. Tujuh jenis kohesi metode telah diidentifikasi (lihat Gambar 8-9). Mulai dari kohesi fungsional (baik) hingga kohesi kebetulan (buruk). Secara umum, kohesi metode harus dimaksimalkan.

Kelas Kohesi adalah tingkat kohesi di antara atribut dan metode kelas (yaitu, seberapa berpikiran tunggal sebuah kelas). Kelas harus mewakili hanya satu hal, seperti karyawan, departemen, atau pesanan. Semua atribut dan metode yang terkandung dalam suatu kelas harus diperlukan agar kelas dapat merepresentasikan benda tersebut. Misalnya, kelas karyawan harus memiliki atribut yang berhubungan dengan nomor jaminan sosial, nama belakang, nama depan, inisial tengah, alamat, dan manfaat, tetapi tidak boleh memiliki atribut seperti pintu, mesin, atau kap mesin. Selanjutnya, tidak boleh ada atribut atau metode yang tidak pernah digunakan. Dengan kata lain, sebuah kelas seharusnya hanya memiliki atribut dan metode yang diperlukan untuk sepenuhnya mendefinisikan contoh untuk masalah yang dihadapi. Dalam hal ini, kita memiliki kohesi kelas yang ideal. Glenford Meyers menyarankan bahwa kelas kohesif harus memiliki atribut berikut:

- Itu harus berisi beberapa metode yang terlihat di luar kelas (yaitu, kelas metode tunggal jarang masuk akal).
- Setiap metode yang terlihat hanya melakukan satu fungsi (yaitu, memiliki kohesi fungsional; lihat Gambar 8-9).
- Semua metode hanya mereferensikan atribut atau metode lain yang didefinisikan dalam kelas atau salah satu superkelasnya (yaitu, jika suatu metode akan mengirim pesan ke objek lain, objek jarak jauh harus berupa nilai dari salah satu atribut objek lokal).
- Seharusnya tidak memiliki coupling kontrol antara metode yang terlihat (lihat Gambar 8-7).

Level	Tipe	Diskripsi
Good	Functional	Sebuah metode untuk melakukan tugas terkait masalah tunggal (misalnya, menghitung IPK saat ini).
	Sequential	Metode ini menggabungkan dua fungsi di mana output yang pertama digunakan sebagai input untuk yang kedua (misalnya, memformat dan memvalidasi IPK saat ini).
	Communicational	Metode ini menggabungkan dua fungsi yang menggunakan atribut yang sama untuk dieksekusi (misalnya, menghitung IPK saat ini dan kumulatif).
	Procedural	Metode ini mendukung beberapa fungsi terkait. Misalnya, metode tersebut dapat menghitung IPK siswa, mencetak catatan siswa, menghitung IPK kumulatif, dan mencetak IPK kumulatif.
	Temporal or Classical	Metode ini mendukung beberapa fungsi terkait dalam waktu (misalnya, menginisialisasi semua atribut).
	Logical	Metode ini mendukung beberapa fungsi terkait, tetapi pilihan fungsi spesifik dipilih berdasarkan variabel kontrol yang dilewatkan ke dalam metode. Misalnya, metode yang dipanggil dapat membuka rekening giro, membuka rekening tabungan, atau menghitung pinjaman, tergantung pada pesan yang dikirim oleh metode pemanggilannya.
Bad	Coincidental	Tujuan metode tidak dapat didefinisikan atau melakukan beberapa fungsi yang tidak terkait satu sama lain. Misalnya, metode ini dapat memperbarui catatan pelanggan, menghitung pembayaran pinjaman, mencetak laporan pengecualian, dan menganalisis struktur harga pesaing.

Sumber: Jenis ini didasarkan pada materi dari Page-Jones, Panduan Praktis untuk Sistem Terstruktur; Myers, Desain Komposit/Terstruktur; Edward Yourdon dan Larry L. Constantine, TerstrukturDesain: Dasar-dasar Disiplin Program Komputer dan Desain Sistem (Englewood Cliffs, NJ: Prentice-Hall, 1979).

Gambar 8-9 Jenis Metode Kohesi

Page-Jones telah mengidentifikasi tiga jenis kohesi kelas yang kurang diinginkan: contoh campuran, domain campuran, dan peran campuran (lihat Gambar 8-10). Sebuah kelas individu dapat memiliki campuran dari salah satu dari tiga jenis.

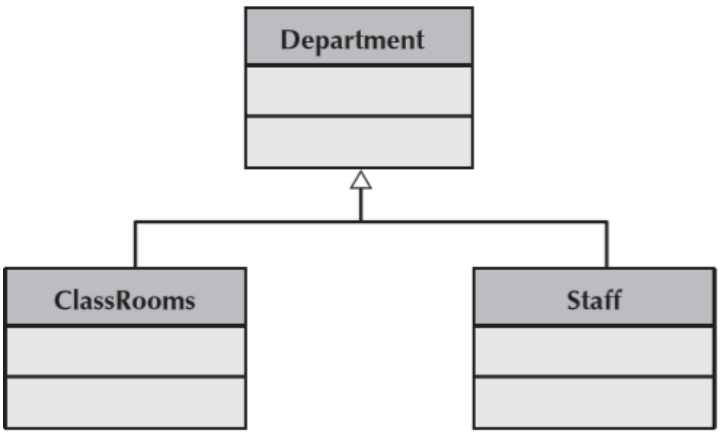
Kohesi generalisasi/spesialisasi membahas kepekaan hierarki pewarisan. Bagaimana kelas-kelas dalam hierarki pewarisan terkait? Apakah kelas terkait melalui semantik generalisasi/spesialisasi (semacam)? Atau, apakah mereka terkait melalui beberapa jenis hubungan asosiasi, agregasi, atau keanggotaan yang dibuat untuk tujuan penggunaan kembali yang sederhana? Ingat semua masalah yang diangkat sebelumnya tentang penggunaan warisan.

Sebagai contoh, pada Gambar 8-11, subclass ClassRooms dan Staff mewarisi dari superclass Department. Jelas, instance dari kelas ClassRooms dan Staff bukanlah sejenis Departemen. Namun, pada hari-hari awal pemrograman berorientasi objek, penggunaan pewarisan ini cukup umum. Ketika seorang programmer melihat bahwa ada beberapa properti umum yang dimiliki oleh sekumpulan kelas, programmer akan membuat abstraksi buatan yang mendefinisikan kesamaan. Ini berpotensi berguna dalam arti penggunaan kembali, tetapi ternyata menyebabkan banyak mimpi buruk pemeliharaan. Dalam hal ini, instance dari ClassRooms dan kelas Staf dikaitkan dengan atau sebagian dari instance Department. Hari ini kita tahu bahwa hierarki pewarisan yang sangat kohesif seharusnya hanya mendukung semantik generalisasi dan spesialisasi (a-kind-of) dan prinsip substitusi.

Level	Tipe	Diskripsi
Good	Ideal	Kelas tidak memiliki kohesi campuran.
	Mixed-Role	Kelas memiliki satu atau lebih atribut yang menghubungkan objek kelas ke objek lain pada lapisan yang sama (misalnya, lapisan domain masalah), tetapi atribut tidak ada hubungannya dengan semantik yang mendasari kelas.
	Mixed-Domain	Kelas memiliki satu atau lebih atribut yang menghubungkan objek kelas dengan objek lain pada lapisan yang berbeda. Dengan demikian, mereka tidak ada hubungannya dengan semantik yang mendasari hal yang diwakili oleh kelas. Dalam kasus ini, atribut yang menyinggung termasuk dalam kelas lain yang terletak di salah satu lapisan lainnya. Misalnya, atribut port yang terletak di kelas domain masalah harus berada di kelas arsitektur sistem yang terkait dengan kelas domain masalah.
Worse	Mixed-Instance	Kelas mewakili dua jenis objek yang berbeda. Kelas harus didekomposisi menjadi dua kelas terpisah. Biasanya, instance yang berbeda hanya menggunakan sebagian dari definisi lengkap kelas.

Berdasarkan materi dari Page-Jones, Fundamentals of Object-Oriented Design di UML.

Gambar 8-10 Jenis-Jenis Kohesi Kelas



Gambar 8-11 Generalisasi/ Spesialisasi vs. Penyalahgunaan Warisan

Connascence

Connascence menggeneralisasi ide-ide kohesi dan coupling, dan menggabungkan mereka dengan argumen untuk enkapsulasi. Untuk mencapai hal ini, tiga tingkat enkapsulasi

telah diidentifikasi. Enkapsulasi level-0 mengacu pada jumlah enkapsulasi yang direalisasikan dalam baris kode individu, enkapsulasi level-1 adalah level enkapsulasi yang dicapai dengan menggabungkan baris kode ke dalam metode, dan enkapsulasi level-2 dicapai dengan membuat kelas yang berisi keduanya metode dan atribut. Kohesi metode dan coupling interaksi terutama membahas enkapsulasi level-1. Kohesi kelas, kohesi generalisasi/spesialisasi, dan coupling pewarisan hanya menangani enkapsulasi level-2. Connascence, sebagai generalisasi kohesi dan coupling, membahas enkapsulasi level-1 dan level-2.

Tapi apa sebenarnya connascence itu? Connascence secara harfiah berarti dilahirkan bersama. Dari perspektif desain berorientasi objek, itu benar-benar berarti bahwa dua modul (kelas atau metode) sangat terkait sehingga jika Anda membuat perubahan di salah satu, kemungkinan besar perubahan yang lain akan diperlukan. Di permukaan, ini sangat mirip dengan coupling dan, dengan demikian, harus diminimalkan. Namun, ketika Anda menggabungkannya dengan level enkapsulasi, itu tidak sesederhana itu. Dalam hal ini, kami ingin meminimalkan connascence keseluruhan dengan menghilangkan connascence yang tidak perlu di seluruh sistem; meminimalkan connascence melintasi batas enkapsulasi apa pun, seperti batas metode dan batas kelas; dan memaksimalkan kedekatan dalam batas enkapsulasi apa pun.

Berdasarkan pedoman ini, subkelas tidak boleh secara langsung mengakses atribut atau metode tersembunyi apa pun dari superkelas [yaitu, subkelas tidak boleh memiliki hak khusus atas properti superkelasnya]. Jika akses langsung ke atribut dan metode yang tidak terlihat dari superclass oleh subclass-nya diizinkan—dan diizinkan di sebagian besar bahasa pemrograman berorientasi objek—dan modifikasi superclass dibuat, maka karena hubungan antara subclass dan superclass-nya, kemungkinan bahwa modifikasi pada subclass juga diperlukan. Dengan kata lain, subclass memiliki akses ke sesuatu melintasi batas enkapsulasi (batas kelas antara subclass dan superclass). Secara praktis, Anda harus memaksimalkan kohesi (connascence) dalam batas enkapsulasi dan meminimalkan coupling (connascence) antara batas enkapsulasi. Ada banyak kemungkinan jenis connascence. Gambar 8-12 menjelaskan lima jenis.

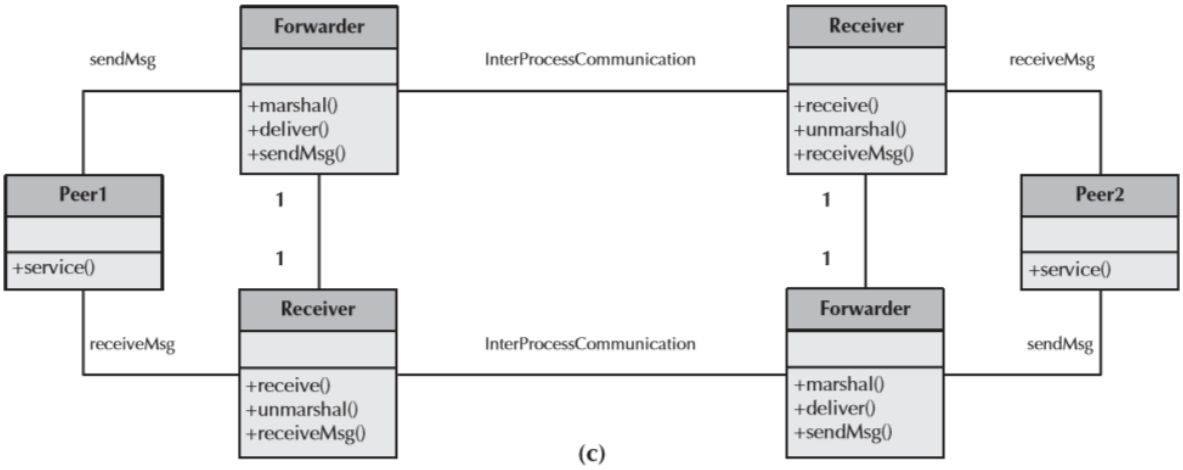
Tipe	Diskripsi
Name	Jika suatu metode mengacu pada suatu atribut, maka metode tersebut terkait dengan nama atribut tersebut. Jika nama atribut berubah, konten metode harus diubah.
Type or Class	Jika sebuah kelas memiliki atribut bertipe A, kelas tersebut terikat dengan tipe atribut tersebut.Jika tipe atribut berubah, deklarasi atribut harus diubah.
Convention	Kelas memiliki atribut di mana rentang nilai memiliki makna semantik (misalnya, nomor akun yang nilainya berkisar dari 1000 hingga 1999 adalah aset).Jika rentang akan berubah, maka setiap metode yang menggunakan atribut tersebut harus dimodifikasi.
Algorithm	Dua metode berbeda dari suatu kelas bergantung pada algoritma yang sama untuk mengeksekusi dengan benar (misalnya, memasukkan elemen ke dalam array dan menemukan elemen dalam array yang sama). Jika algoritma yang mendasarinya akan berubah, maka metode insert dan find juga harus berubah.
Position	Urutan kode dalam suatu metode atau urutan argumen ke suatu metode sangat penting agar metode dapat dieksekusi dengan benar. Jika salah satunya salah, maka metode tersebut, setidaknya, tidak akan berfungsi dengan benar.

Berdasarkan materi dari Meilir Page-Jones, “Comparing Techniques by Means Encapsulation and Connascence” dan Meilir Page Jones, Fundamentals of Object-Oriented Design in UML.

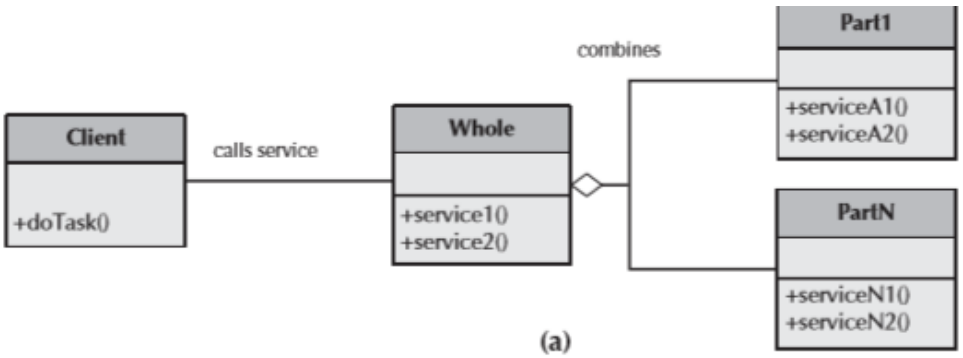
Gambar 8-12 Jenis Connascence

8.5 KEGIATAN DESAIN OBJEK

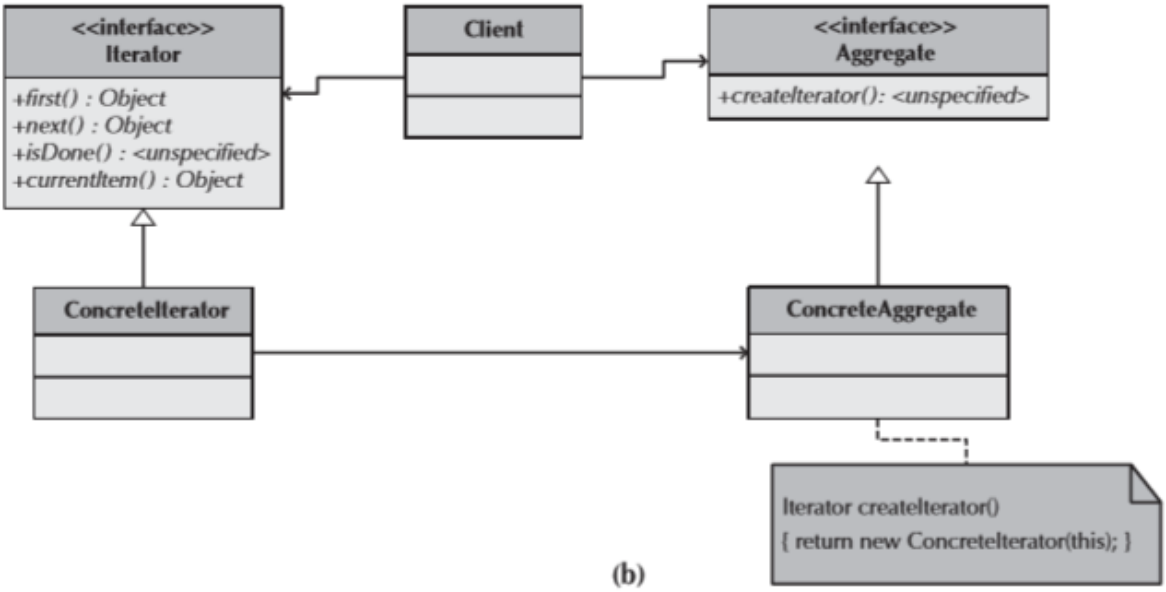
Kegiatan desain untuk kelas dan metode sebenarnya merupakan perpanjangan dari kegiatan analisis dan evolusi yang disajikan sebelumnya (lihat Bab 4 sampai 7). Dalam hal ini, kami memperluas deskripsi partisi, lapisan, dan kelas. Secara praktis, deskripsi yang diperluas dibuat melalui aktivitas yang terjadi selama desain detail kelas dan metode. Kegiatan yang digunakan untuk merancang kelas dan metode termasuk spesifikasi tambahan dari model saat ini, mengidentifikasi peluang untuk digunakan kembali, merestrukturisasi desain, mengoptimalkan desain, dan, akhirnya, memetakan kelas domain masalah ke bahasa implementasi. Tentu saja, setiap perubahan yang dilakukan pada kelas pada satu lapisan dapat menyebabkan kelas pada lapisan lain yang digabungkan dengannya juga dimodifikasi.



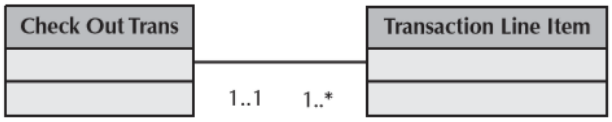
Gambar 8-13 Contoh Pola Desain

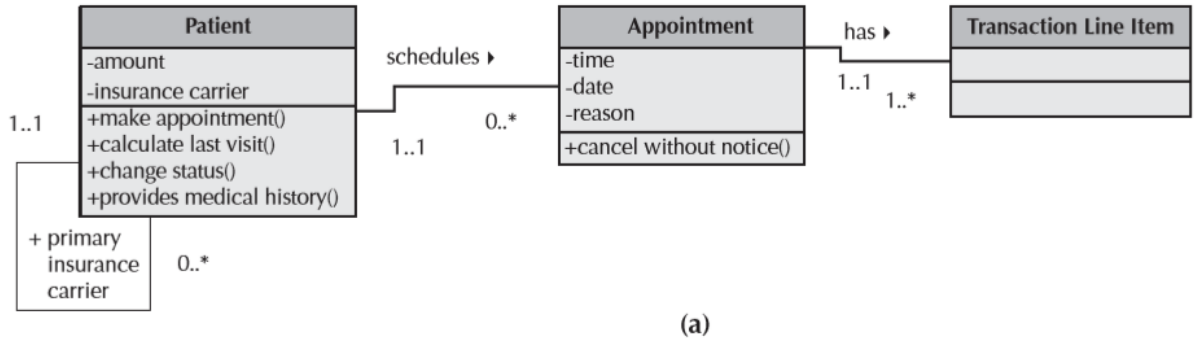

Client
+doTask()
calls service
Whole
+service1()
+service2()
combines
Part1
+serviceA1()
+serviceA2()
PartN
+serviceN1()
+serviceN2()

(a)

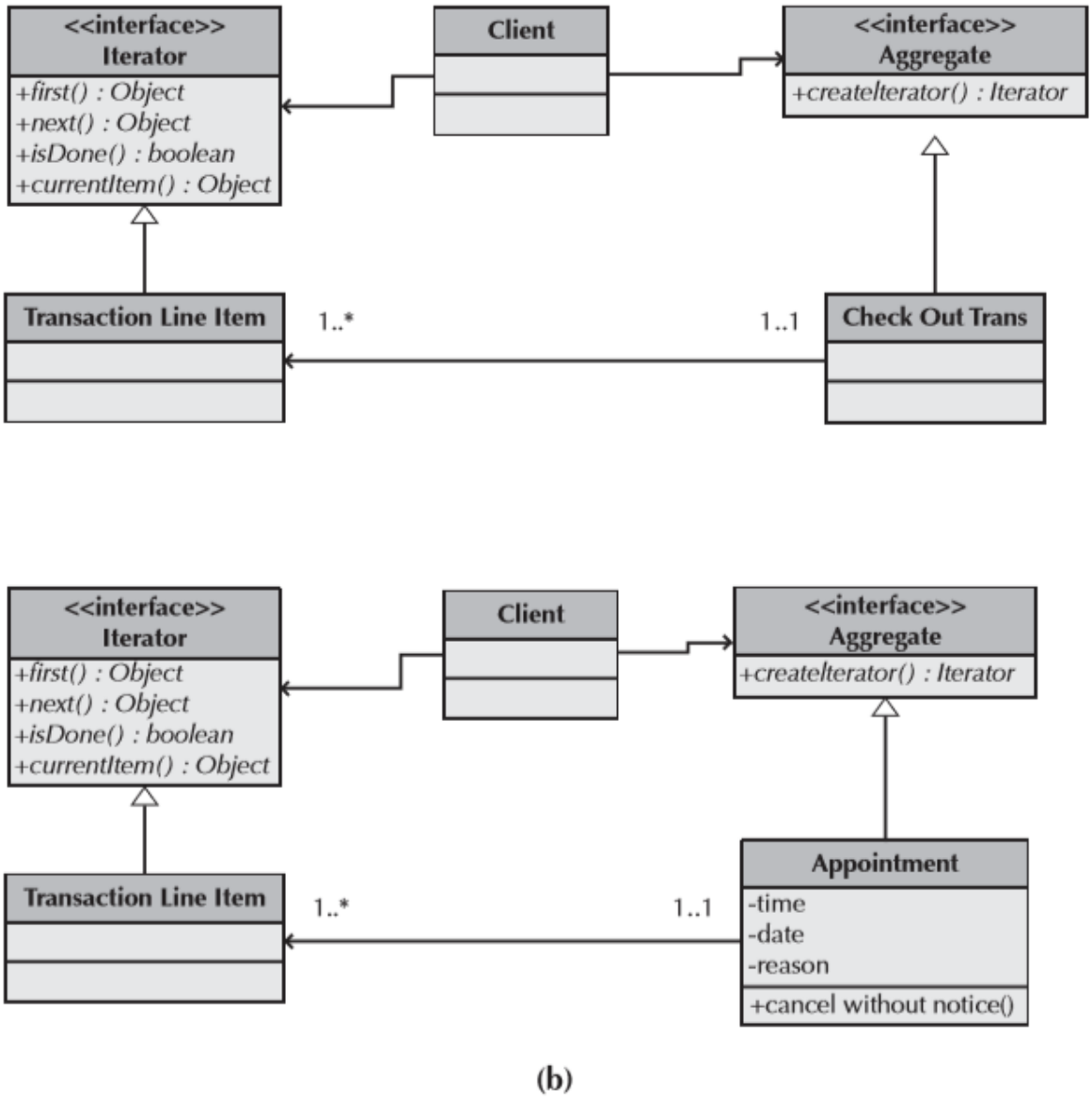

<<interface>>
Iterator
+first() : Object
+next() : Object
+isDone() : <unspecified>
+currentItem() : Object
Client
<<interface>>
Aggregate
+createIterator(): <unspecified>
ConcreteIterator
ConcreteAggregate
Iterator createIterator()
{ return new ConcreteIterator(this); }

(b)


Check Out Trans
Transaction Line Item
1..1
1..*


Patient
-amount
-insurance carrier
+make appointment()
+calculate last visit()
+change status()
+provides medical history()
1..1
+ primary insurance carrier
0..*
schedules
1..1
0..*
Appointment
-time
-date
-reason
+cancel without notice()
has
1..1
1..*
Transaction Line Item

(a)



Gambar 8-14 Pola Desain Iterator yang Diterapkan pada Masalah Library dan Appointment

Menambahkan Spesifikasi

Pada titik ini dalam pengembangan sistem, sangat penting untuk meninjau rangkaian model fungsional, struktural, dan perilaku saat ini. Pertama, kita harus memastikan bahwa kelas pada lapisan domain masalah diperlukan dan cukup untuk memecahkan masalah yang mendasarinya. Untuk melakukan ini, kita perlu memastikan bahwa tidak ada atribut atau metode yang hilang dan tidak ada atribut atau metode tambahan atau tidak terpakai di setiap kelas. Selanjutnya, apakah ada kelas yang hilang atau tambahan? Jika kita telah melakukan pekerjaan kita dengan baik selama analisis, akan ada sedikit, jika ada, atribut, metode, atau kelas untuk ditambahkan ke model. Dan kecil kemungkinannya kita memiliki atribut, metode, atau kelas tambahan untuk dihapus dari model. Namun, kita masih perlu memastikan bahwa kita telah memfaktorkan, mengabstraksi, dan menyempurnakan model yang berkembang dan menciptakan partisi dan kolaborasi yang relevan (lihat Bab 7).

Kedua, kita perlu menyelesaikan visibilitas (tersembunyi atau terlihat) dari atribut dan metode di setiap kelas. Tergantung pada bahasa pemrograman berorientasi objek yang digunakan, ini dapat ditentukan sebelumnya. [Misalnya, di Smalltalk, atribut disembunyikan

dan metode terlihat. Bahasa lain memungkinkan programmer untuk mengatur visibilitas setiap atribut atau metode. Misalnya, di C++ dan Java, Anda dapat mengatur visibilitas ke privat (tersembunyi), publik (terlihat), atau terlindungi (terlihat oleh subkelas, tetapi tidak untuk kelas lain).]22 Secara default, sebagian besar analisis dan desain berorientasi objek pendekatan mengasumsikan pendekatan Smalltalk.

Ketiga, kita perlu memutuskan tanda tangan setiap metode di setiap kelas. Tanda tangan suatu metode terdiri dari tiga bagian: nama metode, parameter atau argumen yang harus diteruskan ke metode, termasuk jenis objeknya, dan jenis nilai yang akan dikembalikan metode ke metode pemanggilan. Tanda tangan suatu metode terkait dengan kontrak metode.

Keempat, kita perlu mendefinisikan batasan apa pun yang harus dipertahankan oleh objek (misalnya, atribut objek yang hanya dapat memiliki nilai dalam rentang tertentu). Ada tiga jenis kendala yang berbeda: prakondisi, pascakondisi, dan invarian. Ini ditangkap dalam bentuk kontrak dan asersi yang ditambahkan ke kartu CRC dan diagram kelas. Kita juga harus memutuskan bagaimana menangani pelanggaran batasan. Haruskah sistem dibatalkan begitu saja? Haruskah sistem secara otomatis membatalkan perubahan yang menyebabkan pelanggaran? Haruskah sistem membiarkan pengguna akhir menentukan pendekatan untuk memperbaiki pelanggaran? Dengan kata lain, perancang harus merancang kesalahan yang diharapkan dapat ditangani oleh sistem. Yang terbaik adalah tidak meninggalkan jenis keputusan desain ini untuk dipecahkan oleh programmer. Pelanggaran batasan dikenal sebagai pengecualian dalam bahasa seperti C^{++} dan Java.

Meskipun kami telah menjelaskan kegiatan ini dalam konteks lapisan domain masalah, mereka juga berlaku untuk lapisan lain: manajemen data (Bab 9), interaksi manusia-komputer (Bab 10), dan arsitektur fisik (Bab 11).

Mengidentifikasi Peluang untuk Digunakan Kembali

Sebelumnya, kami melihat kemungkinan menggunakan kembali model kami dalam analisis melalui penggunaan pola (lihat Bab 5). Dalam desain, selain menggunakan pola analisis, ada peluang untuk menggunakan pola desain, kerangka kerja, pustaka, dan komponen. Peluangnya bervariasi tergantung pada lapisan mana yang sedang ditinjau. Misalnya, diragukan bahwa perpustakaan kelas akan banyak membantu pada lapisan domain masalah, tetapi perpustakaan kelas dapat sangat membantu pada lapisan dasar.

Seperti pola analisis, pola desain hanyalah pengelompokan kelas yang berkolaborasi yang memberikan solusi untuk masalah yang umum terjadi. Perbedaan utama antara analisis dan pola desain adalah bahwa pola desain berguna dalam memecahkan "masalah desain umum dalam konteks tertentu", sedangkan pola analisis cenderung membantu mengisi representasi domain masalah. Misalnya, pola desain yang berguna adalah pola Seluruh Bagian (lihat Gambar 8-13a). Pola Whole-Part secara eksplisit mendukung hubungan Agregasi dan Komposisi dalam UML. Pola desain lain yang berguna adalah pola Iterator (lihat Gambar 8-13b). Tujuan utama dari pola Iterator adalah untuk memberikan pendekatan standar kepada desainer untuk mendukung melintasi berbagai jenis koleksi. Dengan menggunakan pola ini, terlepas dari jenis koleksi (ConcreteAggregate), desainer mengetahui bahwa koleksi perlu membuat iterator (ConcreteIterator) yang menyesuaikan operasi standar yang digunakan untuk melintasi koleksi: first(), next(), isDone( ), dan currentItem(). Mengingat jumlah koleksi yang biasanya ditemukan dalam aplikasi bisnis, pola ini adalah salah satu yang lebih berguna. Misalnya pada Gambar 8-14a, kami mereplikasi sebagian dari masalah Pengangkatan dan Perpustakaan yang dibahas dalam bab-bab sebelumnya, dan pada Gambar 8-14b kami menunjukkan bagaimana pola Iterator dapat diterapkan pada bagian-bagian dari desain mereka yang berkembang. Akhirnya, beberapa pola desain mendukung arsitektur fisik yang

berbeda (lihat Bab 11). Misalnya, pola Forwarder-Receiver (lihat Gambar 8-13c) mendukung arsitektur peer-to-peer. Banyak pola desain tersedia dalam kode sumber C++ atau Java.

Kerangka kerja terdiri dari satu set kelas yang diimplementasikan yang dapat digunakan sebagai dasar untuk mengimplementasikan aplikasi. Sebagian besar kerangka kerja memungkinkan kita membuat subkelas untuk diwarisi dari kelas dalam kerangka kerja. Ada kerangka kerja persistensi objek yang dapat dibeli dan digunakan untuk menambahkan kegigihan ke kelas domain masalah, yang akan membantu pada lapisan manajemen data. Tentu saja, ketika mewarisi dari kelas dalam suatu kerangka kerja, kita menciptakan ketergantungan (yaitu, meningkatkan coupling pewarisan dari subkelas ke superkelas). Oleh karena itu, jika kita menggunakan kerangka kerja dan vendor membuat perubahan pada kerangka kerja, setidaknya kita harus mengkompilasi ulang sistem saat kita meningkatkan ke versi kerangka kerja yang baru.

Pustaka kelas mirip dengan kerangka kerja karena biasanya memiliki seperangkat kelas yang diimplementasikan yang dirancang untuk digunakan kembali. Namun, kerangka kerja cenderung lebih spesifik pada domain. Bahkan, kerangka kerja dapat dibangun menggunakan perpustakaan kelas. Pustaka kelas tipikal dapat dibeli untuk mendukung pemrosesan numerik atau statistik, manajemen file (lapisan manajemen data), atau pengembangan antarmuka pengguna (lapisan interaksi manusia-komputer). Dalam beberapa kasus, instance kelas yang terdapat dalam pustaka kelas dapat dibuat, dan dalam kasus lain, kelas di pustaka kelas dapat diperluas dengan membuat subkelas berdasarkan kelas tersebut. Seperti halnya kerangka kerja, jika kita menggunakan pewarisan untuk menggunakan kembali kelas-kelas di perpustakaan kelas, kita akan mengalami semua masalah yang berhubungan dengan penggabungan warisan dan kelahiran. Jika kita langsung membuat instance kelas di perpustakaan kelas, kita akan membuat ketergantungan antara objek kita dan objek perpustakaan berdasarkan tanda tangan dari metode di objek perpustakaan. Ini meningkatkan coupling interaksi antara objek perpustakaan kelas dan objek kita.

Komponen adalah perangkat lunak yang mandiri dan terenkapsulasi yang dapat dipasang ke sistem untuk menyediakan serangkaian fungsi spesifik yang diperlukan. Saat ini, ada banyak komponen yang tersedia untuk dibeli. Sebuah komponen memiliki API yang terdefinisi dengan baik (antarmuka program aplikasi). API pada dasarnya adalah seperangkat antarmuka metode ke objek yang terkandung dalam komponen. Cara kerja internal komponen tersembunyi di balik API. Komponen dapat diimplementasikan menggunakan perpustakaan dan kerangka kerja kelas. Namun, komponen juga dapat digunakan untuk mengimplementasikan kerangka kerja. Kecuali jika API berubah di antara versi komponen, memutakhirkan ke versi baru biasanya hanya memerlukan penautan kembali komponen ke dalam aplikasi. Dengan demikian, kompilasi ulang biasanya tidak diperlukan.

Manakah dari pendekatan ini yang harus kita gunakan? Itu tergantung pada apa yang kita coba bangun. Secara umum, kerangka kerja sebagian besar digunakan untuk membantu mengembangkan objek pada arsitektur fisik, interaksi manusia-komputer, atau lapisan manajemen data; komponen digunakan terutama untuk menyederhanakan pengembangan objek pada domain masalah dan lapisan interaksi manusia-komputer; dan perpustakaan kelas digunakan untuk mengembangkan kerangka kerja dan komponen dan untuk mendukung lapisan dasar. Apapun pendekatan penggunaan kembali ini yang Anda gunakan, Anda harus ingat bahwa penggunaan kembali membawa banyak manfaat potensial dan kemungkinan masalah. Misalnya, perangkat lunak sebelumnya telah diverifikasi dan divalidasi, yang seharusnya mengurangi jumlah pengujian yang diperlukan untuk sistem kami. Namun seperti yang dinyatakan sebelumnya, jika perangkat lunak yang menjadi basis sistem kami berubah, maka kemungkinan besar, kami juga harus mengubah sistem kami. Selanjutnya, jika perangkat lunak berasal dari perusahaan pihak ketiga, kami menciptakan ketergantungan dari

perusahaan kami (atau perusahaan klien kami) ke vendor pihak ketiga. Akibatnya, kita perlu memiliki keyakinan bahwa vendor akan berada dalam bisnis untuk sementara waktu.

Restrukturisasi Desain

Setelah kelas dan metode individu telah ditentukan dan perpustakaan kelas, kerangka kerja, dan komponen telah dimasukkan ke dalam desain yang berkembang, kita harus menggunakan pemfaktoran untuk merestrukturisasi desain. Pemfaktoran (Bab 7) adalah proses memisahkan aspek-aspek dari suatu metode atau kelas menjadi metode atau kelas baru untuk menyederhanakan desain keseluruhan. Misalnya, ketika meninjau satu set kelas pada lapisan tertentu, kita mungkin menemukan bahwa subset dari kelas tersebut memiliki definisi yang sama. Dalam hal ini, mungkin berguna untuk memfaktorkan kesamaan dan membuat kelas baru. Berdasarkan isu-isu yang berkaitan dengan kohesi, coupling, dan connascence, kelas baru dapat dikaitkan dengan kelas lama melalui pewarisan (generalisasi) atau melalui hubungan agregasi atau asosiasi.

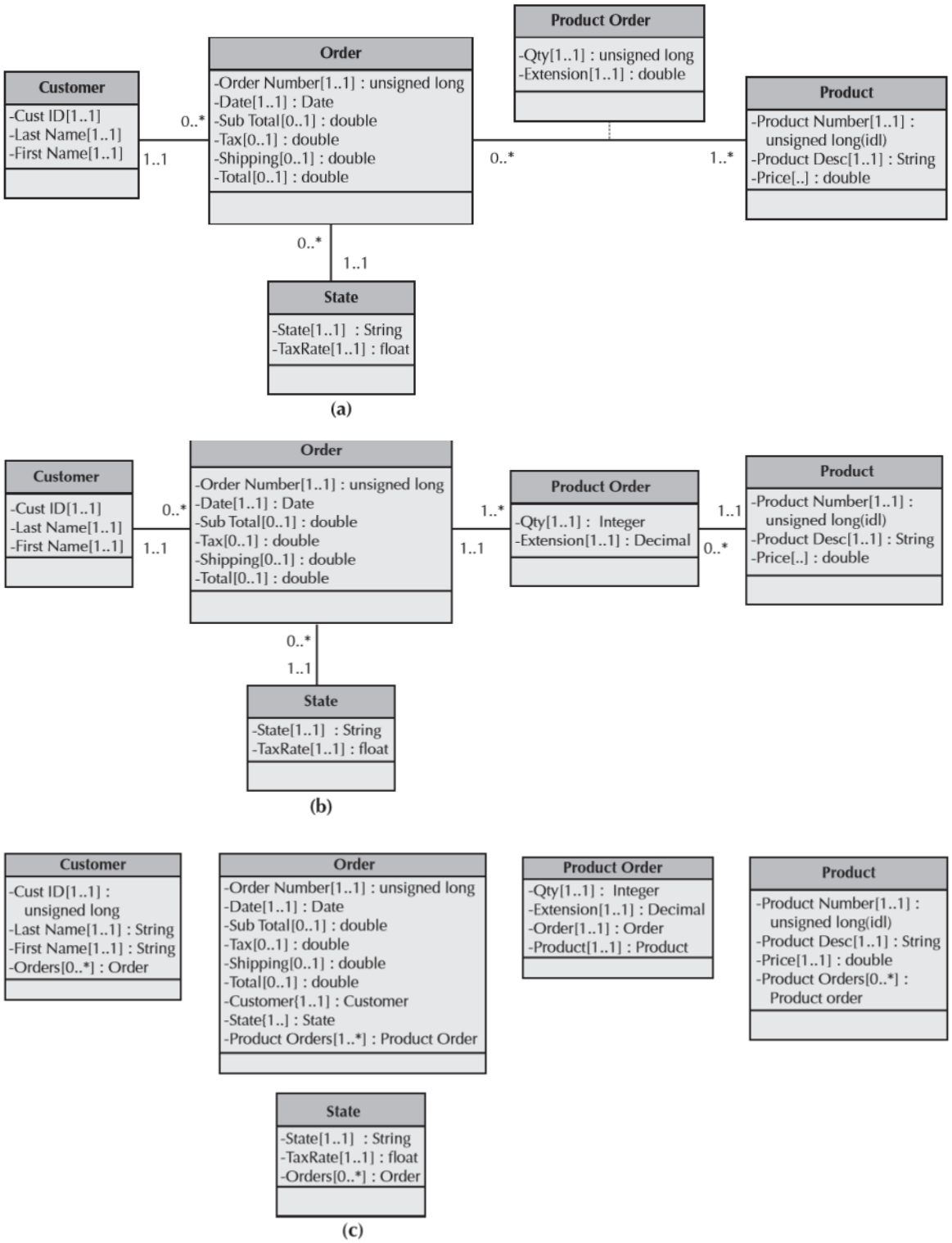
Proses lain yang berguna untuk merestrukturisasi desain yang berkembang adalah normalisasi. Normalisasi dijelaskan dalam Bab 9 dalam kaitannya dengan database relasional. Namun, normalisasi terkadang berguna untuk mengidentifikasi kelas potensial yang hilang dari desain. Juga terkait dengan normalisasi adalah persyaratan untuk menerapkan hubungan asosiasi dan agregasi yang sebenarnya sebagai atribut. Hampir tidak ada bahasa pemrograman berorientasi objek yang membedakan antara atribut dan hubungan asosiasi dan agregasi. Oleh karena itu, semua hubungan asosiasi dan agregasi harus dikonversi ke atribut di kelas. Misalnya pada Gambar 8-15a, kelas Pelanggan dan Negara dikaitkan dengan kelas Pesanan. Selanjutnya, kelas asosiasi Product-Order diasosiasikan dengan kelas Order dan Product. Salah satu hal pertama yang harus dilakukan adalah mengubah kelas Asosiasi Pesanan Produk menjadi kelas normal. Perhatikan nilai multiplisitas untuk asosiasi baru antara kelas Pesanan dan Pesanan Produk serta kelas Pesanan Produk dan Produk (lihat Gambar 8-15b). Selanjutnya, kita perlu mengonversi semua asosiasi menjadi atribut yang mewakili hubungan antara kelas yang terpengaruh. Dalam hal ini, kelas Pelanggan harus memiliki atribut Pesanan yang ditambahkan untuk mewakili kumpulan pesanan yang mungkin dimiliki oleh turunan kelas Pelanggan; kelas Pesanan harus menambahkan atribut ke instance referensi dari kelas Pelanggan, Negara Bagian, dan Pesanan Produk; kelas Negara harus memiliki atribut yang ditambahkan untuk referensi semua contoh kelas Orde yang terkait dengan negara tertentu; kelas Pesanan Produk baru harus memiliki atribut yang memungkinkan turunan dari kelas Pesanan Produk untuk mereferensikan turunan mana dari kelas Pesanan dan turunan mana dari kelas Produk yang relevan dengannya; dan, akhirnya, kelas Produk harus menambahkan atribut yang mereferensikan instance yang relevan dari kelas Pesanan Produk (lihat Gambar 8-15c). Seperti yang Anda lihat, bahkan dalam contoh yang sangat kecil ini, banyak perubahan yang perlu dilakukan untuk menyiapkan desain untuk implementasi.

Akhirnya, semua hubungan pewarisan harus ditantang untuk memastikan bahwa mereka hanya mendukung semantik generalisasi/spesialisasi (sejenis). Jika tidak, semua masalah yang disebutkan sebelumnya dengan penggabungan pewarisan, kohesi kelas, dan kohesi generalisasi/spesialisasi akan terjadi.

Optimisasi Desain

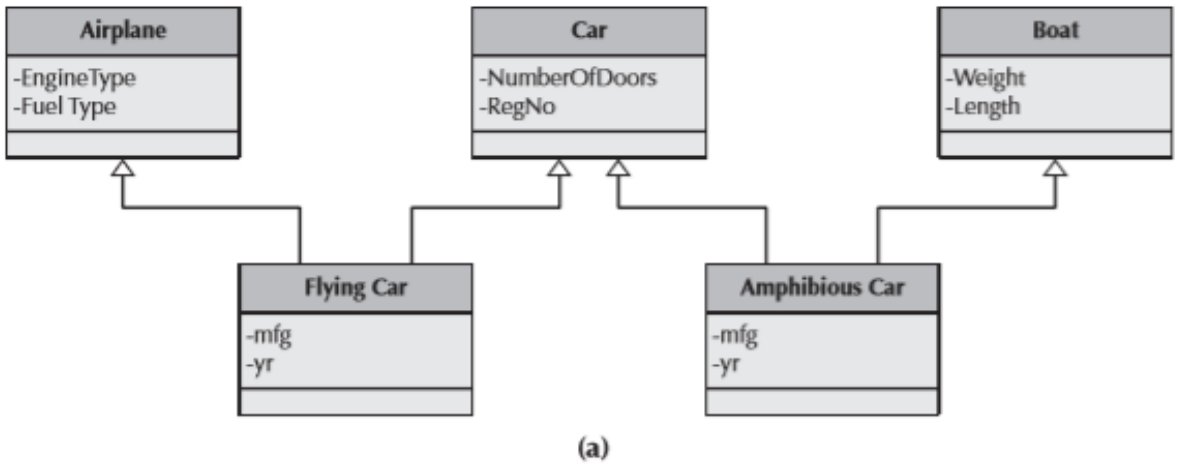
Sampai sekarang, kami telah memfokuskan energi kami untuk mengembangkan desain yang dapat dimengerti. Dengan semua kelas, pola, kolaborasi, partisi, dan lapisan yang dirancang dan dengan semua perpustakaan kelas, kerangka kerja, dan komponen yang disertakan dalam desain, pemahaman telah menjadi fokus utama kami. Namun, meningkatkan pemahaman desain biasanya menciptakan desain yang tidak efisien. Sebaliknya, fokus pada

masalah efisiensi akan menghasilkan desain yang lebih sulit untuk dipahami. Desain praktis yang baik mengelola pertukaran tak terelakkan yang harus terjadi.

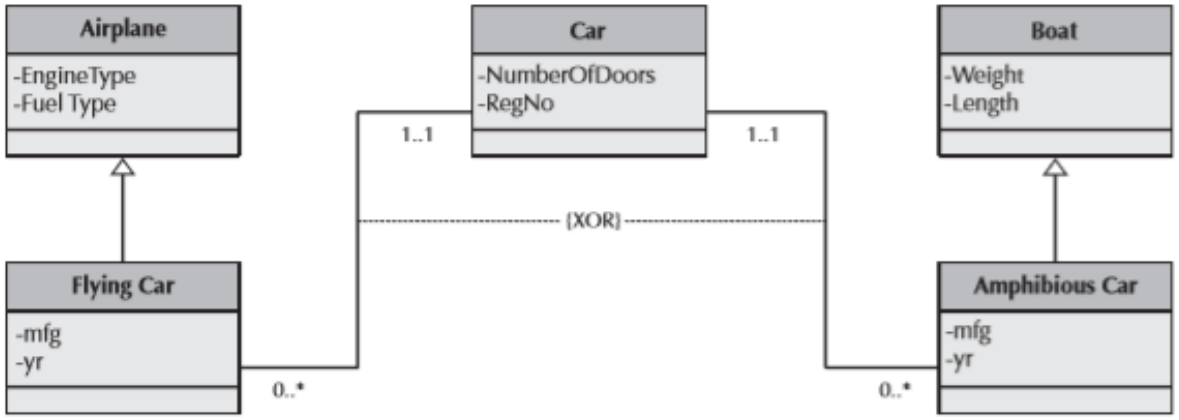


Gambar 8-15 Mengonversi Asosiasi menjadi Atribut

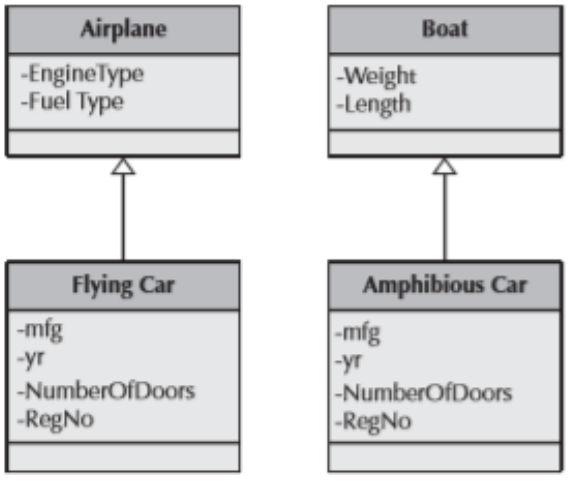
Optimalisasi pertama yang perlu dipertimbangkan adalah meninjau jalur akses antar objek. Dalam beberapa kasus, pesan dari satu objek ke objek lain memiliki jalur panjang untuk dilalui (yaitu, melewati banyak objek). Jika jalur panjang dan pesan sering dikirim, jalur redundan harus dipertimbangkan. Menambahkan atribut ke objek panggilan yang akan menyimpan koneksi langsung ke objek di akhir jalur dapat menyelesaikan ini.



(a)



(b)



Gambar 8-16 Memfaktorkan Efek Pewarisan Ganda untuk Bahasa dengan Pewarisan Tunggal

Optimalisasi kedua adalah meninjau setiap atribut dari setiap kelas. Harus ditentukan metode mana yang menggunakan atribut dan objek mana yang menggunakan metode. Jika satu-satunya metode yang menggunakan atribut adalah metode baca dan perbarui dan hanya instance dari satu kelas yang mengirim pesan untuk membaca dan memperbarui atribut, maka atribut tersebut mungkin termasuk dalam kelas pemanggil, bukan kelas yang dipanggil. Memindahkan atribut ke kelas panggilan secara substansial akan mempercepat sistem.

Optimalisasi ketiga adalah meninjau penyebaran langsung dan tidak langsung dari setiap metode. Fan-out mengacu pada jumlah pesan yang dikirim oleh suatu metode. Penyebaran langsung adalah jumlah pesan yang dikirim oleh metode itu sendiri, sedangkan penyebaran tidak langsung juga mencakup jumlah pesan yang dikirim oleh metode yang dipanggil oleh metode lain dalam pohon pesan. Jika penyebaran suatu metode relatif tinggi terhadap metode lain dalam sistem, metode tersebut harus dioptimalkan. Salah satu cara untuk melakukannya adalah dengan mempertimbangkan untuk menambahkan indeks ke atribut yang digunakan untuk mengirim pesan ke objek di pohon pesan.

Optimalisasi keempat adalah dengan melihat urutan eksekusi pernyataan dalam metode yang sering digunakan. Dalam beberapa kasus, dimungkinkan untuk mengatur ulang beberapa pernyataan agar lebih efisien. Misalnya, jika berdasarkan objek dalam sistem diketahui bahwa rutinitas pencarian dapat dipersempit dengan mencari satu atribut sebelum atribut lainnya, maka algoritma pencarian harus dioptimalkan dengan memaksanya untuk selalu mencari dalam urutan yang telah ditentukan.

Optimalisasi kelima adalah menghindari penghitungan ulang dengan membuat atribut turunan (atau nilai aktif) (misalnya, total yang menyimpan nilai komputasi). Hal ini juga dikenal sebagai caching hasil komputasi, dan dapat dicapai dengan menambahkan pemicu ke atribut yang terkandung dalam komputasi (yaitu, atribut yang atribut turunannya bergantung). Ini akan membutuhkan penghitungan ulang untuk dilakukan hanya ketika salah satu atribut yang masuk ke dalam perhitungan diubah. Pendekatan lain adalah dengan menandai atribut turunan untuk penghitungan ulang dan menunda penghitungan ulang hingga atribut turunan diakses berikutnya. Pendekatan terakhir ini menunda penghitungan ulang selama mungkin. Dengan cara ini, perhitungan tidak terjadi kecuali jika itu harus terjadi. Jika tidak, setiap kali atribut turunan perlu diakses, perhitungan akan diperlukan.

Optimalisasi keenam yang harus dipertimbangkan berkaitan dengan objek yang berpartisipasi dalam asosiasi satu-ke-satu; yaitu, keduanya harus ada agar keduanya ada. Dalam hal ini, mungkin masuk akal, untuk tujuan efisiensi, untuk menciutkan dua kelas yang menentukan menjadi satu kelas. Namun, pengoptimalan ini mungkin perlu dipertimbangkan kembali saat menyimpan objek "lebih gemuk" dalam database. Bergantung pada jenis persistensi objek yang digunakan (lihat Bab 9), sebenarnya bisa lebih efisien untuk memisahkan kedua kelas. Sebagai alternatif, akan lebih masuk akal jika kedua kelas digabungkan pada lapisan domain masalah tetapi tetap terpisah pada lapisan manajemen data.

Memetakan Kelas Domain Masalah ke Bahasa Implementasi

Sampai saat ini, telah diasumsikan bahwa kelas dan metode dalam model akan diimplementasikan secara langsung dalam bahasa pemrograman berorientasi objek. Namun, sekarang penting untuk memetakan desain saat ini dengan kemampuan bahasa pemrograman yang digunakan. Misalnya, jika kita telah menggunakan pewarisan berganda dalam desain kita tetapi kita mengimplementasikannya dalam bahasa yang hanya mendukung pewarisan tunggal, maka pewarisan berganda harus difaktorkan di luar desain. Jika implementasi dilakukan dalam bahasa berbasis objek, yang tidak mendukung pewarisan, atau bahasa berbasis non-objek, seperti C, kita harus memetakan objek domain masalah ke konstruksi pemrograman yang dapat diimplementasikan menggunakan lingkungan implementasi yang dipilih.

Menerapkan Kelas Domain Masalah dalam Bahasa Warisan Tunggal. Satu-satunya masalah yang terkait dengan penerapan objek domain masalah adalah pemfaktoran dari beberapa pewarisan berganda—yaitu, penggunaan lebih dari satu superclass—yang digunakan dalam desain yang berkembang. Misalnya, jika Anda menerapkan solusi di Java,

Smalltalk, atau Visual Basic.net, Anda harus memfaktorkan beberapa pewarisan. Cara termudah untuk melakukannya adalah dengan menggunakan aturan berikut:

ATURAN 1a:

> Mengubah hubungan pewarisan tambahan menjadi hubungan asosiasi. Multiplisitas dari asosiasi baru dari subclass ke superclass harus 1..1. Jika superclass tambahan adalah beton, yaitu mereka dapat dipakai sendiri, maka multiplisitas dari superclass ke subclass adalah 0.1. Jika tidak, itu adalah 1.1. Selanjutnya, batasan eksklusif-atau (XOR) harus ditambahkan di antara asosiasi. Terakhir, Anda harus menambahkan metode yang sesuai untuk memastikan bahwa semua informasi masih tersedia untuk kelas asli.

atau

ATURAN 1b:

> Latten hierarki pewarisan dengan menyalin atribut dan metode dari superclass tambahan ke semua subclass dan menghapus superclass tambahan dari desain.

Gambar 8-16 menunjukkan penerapan aturan-aturan ini. Gambar 8-16a menggambarkan contoh sederhana pewarisan berganda di mana Mobil Terbang mewarisi dari Pesawat dan Mobil, dan Mobil Amfibi mewarisi dari Mobil dan Perahu. Dengan asumsi Mobil itu beton, kami menerapkan Aturan 1a ke bagian a, dan kami berakhir dengan diagram di bagian b, di mana kami telah menambahkan asosiasi antara Mobil Terbang dan Mobil dan hubungan antara Mobil Amfibi dan Perahu. Multiplisitas telah ditambahkan dengan benar, dan batasan XOR telah diterapkan. Jika kita menerapkan Aturan 1b ke bagian a, kita berakhir dengan diagram di bagian c, di mana semua atribut Mobil telah disalin ke dalam Mobil Terbang dan Mobil Amfibi. Dalam kasus terakhir ini, Anda mungkin harus berurusan dengan efek konflik pewarisan (lihat sebelumnya di bab ini).

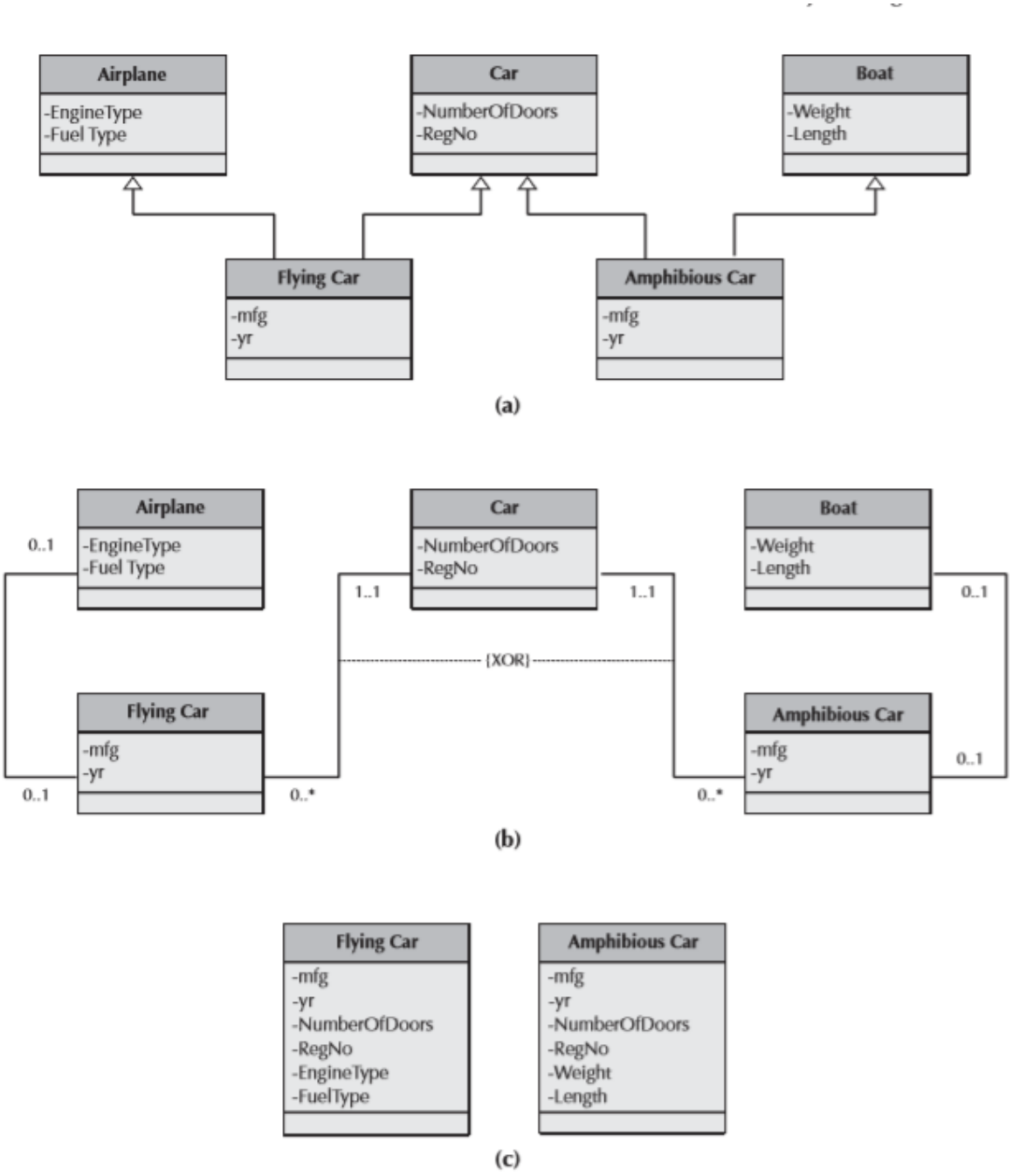
Keuntungan dari Aturan 1a adalah bahwa semua kelas domain masalah yang diidentifikasi selama analisis dipertahankan. Hal ini memungkinkan fleksibilitas maksimum pemeliharaan desain lapisan domain masalah. Namun, Aturan 1a meningkatkan jumlah pengiriman pesan yang diperlukan dalam sistem, dan telah menambahkan persyaratan pemrosesan yang melibatkan batasan XOR, sehingga mengurangi efisiensi desain secara keseluruhan. Oleh karena itu, rekomendasi kami adalah membatasi Aturan 1a untuk diterapkan hanya ketika berhadapan dengan superclass "ekstra" yang konkret karena mereka memiliki keberadaan independen dalam domain masalah. Gunakan Aturan 1b ketika mereka abstrak karena mereka tidak memiliki keberadaan independen dari subkelas.

Menerapkan Objek Domain Masalah dalam Bahasa Berbasis Objek Jika kita akan mengimplementasikan solusi kita dalam bahasa berbasis objek (yaitu, bahasa yang mendukung pembuatan objek tetapi tidak mendukung pewarisan implementasi), kita harus memperhitungkan semua penggunaan warisan dari desain kelas domain masalah. Menerapkan aturan sebelumnya ke semua superclass memungkinkan kita untuk merestrukturisasi desain kita tanpa pewarisan apa pun.

Gambar 8-17 menunjukkan penerapan aturan sebelumnya. Gambar 8-17a menunjukkan contoh sederhana yang sama dari pewarisan berganda yang digambarkan pada Gambar 8-16, di mana Mobil Terbang mewarisi dari Pesawat dan Mobil, dan Mobil Amfibi mewarisi dari Mobil dan Perahu. Dengan asumsi bahwa Pesawat, Mobil, dan Perahu adalah beton, kami menerapkan Aturan 1a untuk bagian a dan kami berakhir dengan diagram di bagian b, di mana kami telah menambahkan asosiasi, multiplisitas, dan kendala XOR. Jika

kita menerapkan Aturan 1b ke bagian a, kita berakhir dengan diagram di bagian c, di mana semua atribut superclass telah disalin ke dalam Mobil Terbang dan Mobil Amfibi. Dalam kasus terakhir ini, Anda mungkin harus berurusan dengan efek konflik warisan.

Menerapkan Objek Domain Masalah dalam Bahasa Tradisional. Dari perspektif praktis, kita jauh lebih baik mengimplementasikan desain berorientasi objek dalam bahasa pemrograman berorientasi objek, seperti C++, Java, Objective-C, atau Visual Basic.net. Secara praktis, jurang pemisah antara desain berorientasi objek dan bahasa pemrograman tradisional terlalu besar untuk dapat dilewati oleh manusia biasa. Saran terbaik yang dapat kami berikan tentang menerapkan desain berorientasi objek dalam bahasa pemrograman tradisional adalah melarikan diri secepat dan sejauh mungkin dari proyek. Namun, jika kita cukup berani (bodoh?) untuk mencoba ini, kita harus menyadari bahwa selain memfaktorkan warisan dari desain, kita harus memfaktorkan semua penggunaan polimorfisme, pengikatan dinamis, enkapsulasi, dan penyembunyian informasi. Ini adalah pekerjaan tambahan yang harus diselesaikan. Cara kami memfaktorkan fitur-fitur berorientasi objek ini dari desain detail sistem cenderung bergantung pada bahasa. Ini di luar cakupan teks ini.



Gambar 8-17 Memfaktorkan Efek Pewarisan Berganda untuk Bahasa Berbasis Objek

8.6 KENDALA DAN KONTRAK

Kontrak diperkenalkan di Bab 5 terkait dengan kolaborasi. Kontrak memformalkan interaksi antara objek klien dan server, di mana objek klien (konsumen) adalah turunan dari kelas yang mengirim pesan ke objek server (pemasok) yang mengeksekusi salah satu metodenya sebagai respons terhadap permintaan. Kontrak dimodelkan pada gagasan hukum kontrak, di mana kedua belah pihak, klien dan objek server, memiliki kewajiban dan hak. Secara praktis, kontrak adalah seperangkat batasan dan jaminan. Jika kendala terpenuhi, maka objek server menjamin perilaku tertentu. Batasan dapat ditulis dalam bahasa alami (misalnya, Inggris), bahasa semiformal (misalnya, Structured English), atau bahasa formal (misalnya, UML's Object Constraint Language).). Mengingat kebutuhan akan spesifikasi kendala yang tepat dan tidak ambigu, kami merekomendasikan penggunaan Bahasa Batasan Objek UML.

Object Constraint Language (OCL) adalah bahasa lengkap yang dirancang untuk menentukan batasan. Di bagian ini, kami memberikan gambaran singkat tentang beberapa konstruksi yang lebih berguna yang terkandung dalam bahasa (lihat Gambar 8-18). Pada dasarnya, semua ekspresi OCL hanyalah pernyataan deklaratif yang mengevaluasi apakah benar atau salah. Jika ekspresi bernilai true, maka kendala telah terpenuhi. Misalnya, jika pelanggan harus memiliki saldo kurang dari seratus dolar untuk diizinkan menempatkan pesanan kredit lain, ekspresi OCL akan menjadi:

saldo terutang <= 100.00

OCL juga memiliki kemampuan untuk melintasi hubungan antar objek, misalnya, jika jumlah pada pesanan pembelian diperlukan untuk menjadi jumlah nilai dari baris pesanan pembelian individu, ini dapat dimodelkan sebagai:

jumlah = OrderLine.sum(getPrice())

OCL juga menyediakan kemampuan untuk memodelkan batasan yang lebih kompleks dengan sekumpulan operator logika: dan, atau, xor, dan bukan. Misalnya, jika pelanggan diberikan diskon hanya jika mereka adalah warga lanjut usia atau pelanggan "utama", OCL dapat digunakan untuk memodelkan kendala sebagai:

usia > 65 atau customerType = “prima”

OCL menyediakan banyak konstruksi lain yang dapat digunakan untuk membangun batasan unik. Ini termasuk operator berorientasi matematika, operator string, dan operator traversal hubungan. Misalnya, jika nama yang dicetak pada pesanan pelanggan harus merupakan gabungan dari nama depan dan nama belakang pelanggan, maka OCL dapat mewakili batasan ini sebagai:

printName = firstName.concat(lastName)

Kita telah melihat contoh operator '.' yang digunakan untuk melintasi hubungan dari Order ke OrderLine di atas. Operator '::' memungkinkan pemodelan melintasi hubungan pewarisan.

OCL juga menyediakan satu set operasi yang digunakan untuk mendukung kendala atas kumpulan objek. Misalnya, kami mendemonstrasikan penggunaan operator sum() di atas di mana kami ingin menjamin bahwa jumlahnya sama dengan penjumlahan semua harga item dalam koleksi. Operasi ukuran mengembalikan jumlah item dalam koleksi. Operasi hitung mengembalikan jumlah kemunculan dalam kumpulan objek tertentu yang diteruskan sebagai argumennya. Operasi penyertaan menguji apakah objek yang diteruskan ke sana sudah termasuk dalam koleksi. Operasi isEmpty menentukan apakah koleksi kosong atau tidak. Operasi pilih menyediakan dukungan untuk memodelkan identifikasi subset dari koleksi berdasarkan ekspresi yang dilewatkan sebagai argumennya. Jelas, OCL menyediakan seperangkat operator dan operasi yang kaya untuk memodelkan kendala.

Operator Type	Operator	Example
Comparison	=	a = 5
	<	a < 100
	<=	a <= 100
	>	a > 100
	>=	a > = 100
	<>	a <> 100
Logical	and	a and b
	or	a or b
	xor	a xor b
	not	not a
Math	+	a + b
	−	a − b
	*	a * b
	/	a / b
String	concat	a = b.concat(c)
Relationship Traversal	.	relationshipAttributeName.b
	::	superclassName::propertyName
Collection	size	a.size
	count(object)	a.count(b)
	includes(object)	a.includes(b)
	isEmpty	a.isEmpty
	sum()	a.sum(b,c,d)
	select(expression)	a.select(b > d)

Gambar 8-18 Contoh Konstruksi OCL

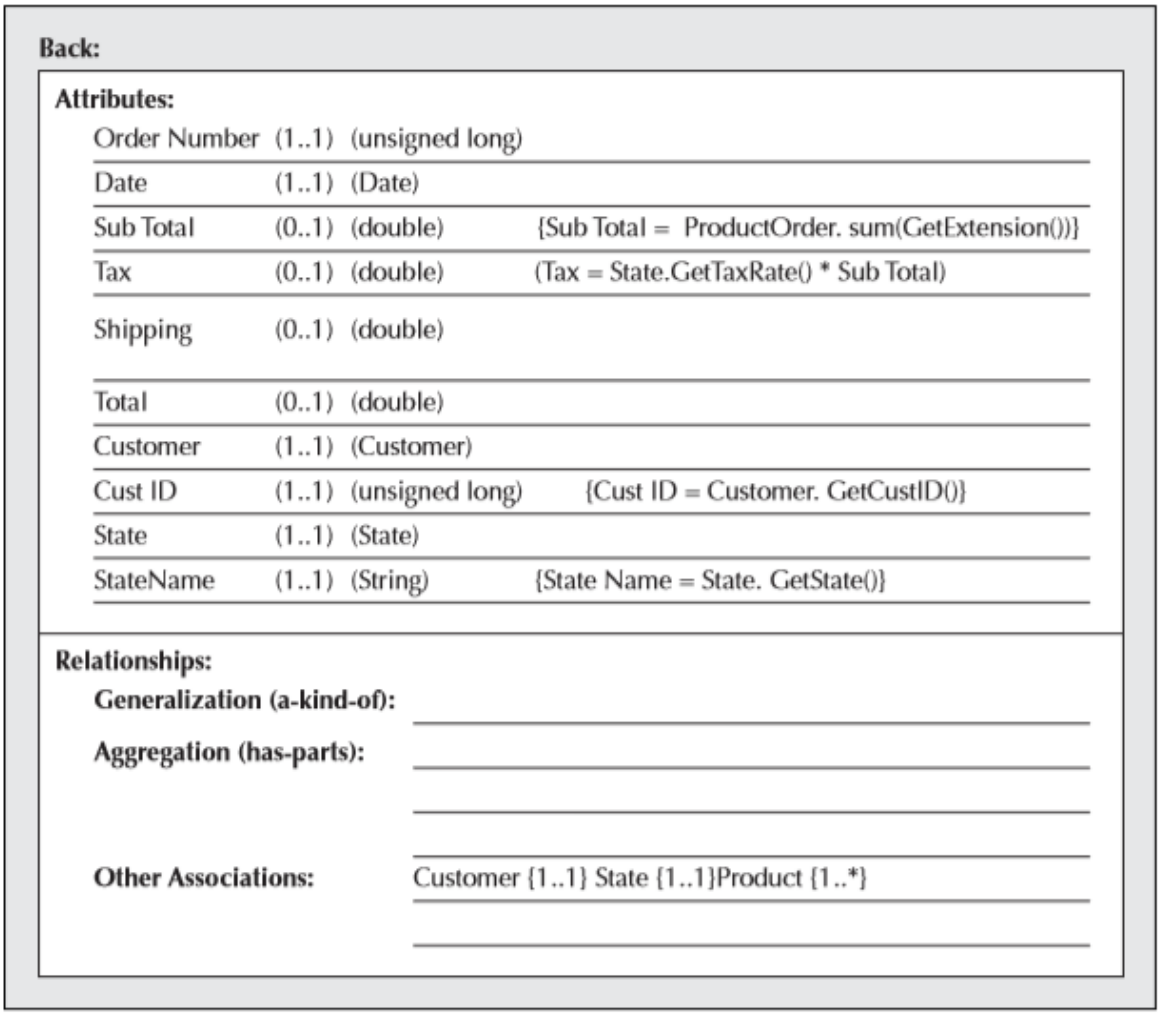
Jenis Kendala

Tiga jenis kendala biasanya ditangkap dalam desain berorientasi objek: prasyarat, pascakondisi, dan invarian.

Kontrak digunakan terutama untuk menetapkan prakondisi dan pascakondisi agar suatu metode dapat dieksekusi dengan benar. Prasyarat adalah batasan yang harus dipenuhi agar metode dapat dieksekusi. Misalnya, parameter yang diteruskan ke metode harus valid agar metode dapat dieksekusi. Jika tidak, pengecualian harus diajukan. Postcondition adalah batasan yang harus dipenuhi setelah metode dieksekusi, atau efek eksekusi metode harus dibatalkan. Misalnya, metode tidak dapat membuat atribut apa pun dari objek mengambil nilai yang tidak valid. Dalam hal ini, pengecualian harus dimunculkan, dan efek eksekusi metode harus dibatalkan.

Sedangkan prekondisi dan postkondisi memodelkan kendala pada metode individu, kendala model invarian yang harus selalu benar untuk semua contoh kelas. Contoh invarian termasuk domain atau jenis atribut, multiplisitas atribut, dan nilai atribut yang valid. Ini termasuk atribut yang memodelkan hubungan asosiasi dan agregasi. Misalnya, jika hubungan asosiasi diperlukan, invarian harus dibuat yang akan memaksanya untuk memiliki nilai yang valid agar instance ada. Invarian biasanya dilampirkan ke kelas. Kita dapat melampirkan

invarian ke kartu CRC atau diagram kelas dengan menambahkan satu set pernyataan kepada mereka.



Back:

Attributes:

Order Number	(1..1)	(unsigned long)	
Date	(1..1)	(Date)	
Sub Total	(0..1)	(double)	{Sub Total = ProductOrder. sum(GetExtension())}
Tax	(0..1)	(double)	(Tax = State.GetTaxRate() * Sub Total)
Shipping	(0..1)	(double)	
Total	(0..1)	(double)	
Customer	(1..1)	(Customer)	
Cust ID	(1..1)	(unsigned long)	{Cust ID = Customer. GetCustID()}
State	(1..1)	(State)	
StateName	(1..1)	(String)	{State Name = State. GetState()}

Relationships:

Generalization (a-kind-of):

Aggregation (has-parts):

Other Associations: Customer {1..1} State {1..1}Product {1..*}

(b)

Gambar 8-19 Invarian pada Kartu CRC

Pada Gambar 8-19, bagian belakang kartu CRC membatasi atribut Order ke tipe tertentu. Misalnya, Nomor Pesanan harus berupa unsigned long, dan Pelanggan harus menjadi turunan dari kelas Pelanggan. Selanjutnya, invarian tambahan ditambahkan ke empat atribut. Misalnya, ID Cust tidak hanya harus berupa unsigned long, tetapi juga harus memiliki satu dan hanya satu nilai [yaitu, multiplisitas (1..1)], dan harus memiliki nilai yang sama dengan hasil GetCustID () pesan yang dikirim ke instance Pelanggan yang disimpan dalam atribut Pelanggan. Juga ditampilkan batasan untuk keberadaan sebuah instance, sebuah instance dari kelas Customer, sebuah instance dari kelas State, dan setidaknya satu instance dari kelas Product harus diasosiasikan dengan objek Order (lihat bagian Hubungan dari kartu CRC di mana kelipatannya masing-masing adalah 1..1, 1..1, dan 1..*). Gambar 8-20 menggambarkan himpunan invarian yang sama pada diagram kelas. Namun, jika semua invarian ditempatkan pada diagram kelas, diagram menjadi sangat sulit untuk dipahami. Akibatnya, kami merekomendasikan untuk memperluas kartu CRC untuk mendokumentasikan invarian daripada melampirkan semuanya ke diagram kelas atau membuat dokumen teks terpisah yang berisi mereka (lihat Gambar 8-21).

Elemen Kontrak

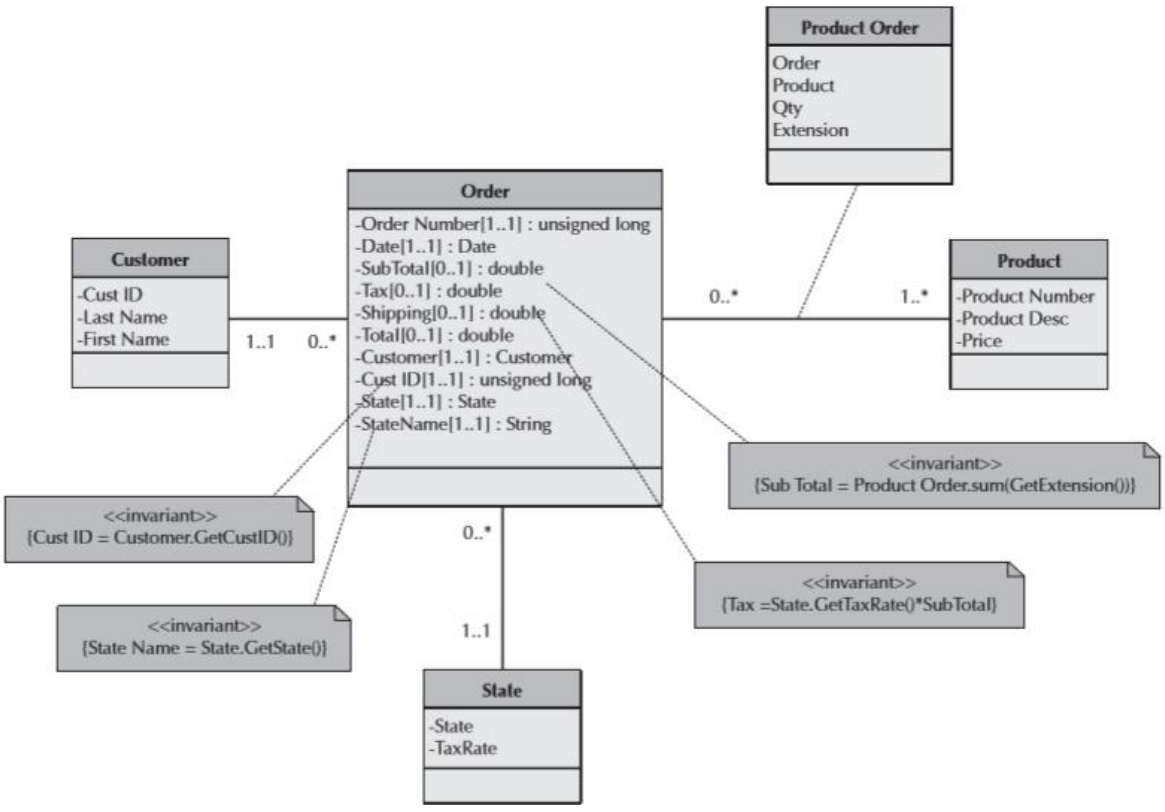
Kontrak mendokumentasikan pengiriman pesan yang terjadi di antara objek. Secara teknis, kontrak harus dibuat untuk setiap pesan yang dikirim dan diterima oleh setiap objek, satu untuk setiap interaksi. Namun, akan ada sedikit duplikasi jika ini dilakukan. Dalam praktiknya, kontrak dibuat untuk setiap metode yang dapat menerima pesan dari objek lain (yaitu, satu untuk setiap metode yang terlihat).

Front:

Class Name: Order	**ID:** 2	**Type:** Concrete, Domain
Description: An Individual who needs to receive or has received medical attention		**Associated Use Cases:** 3

Responsibilities	Collaborators
Calculate subtotal	
Calculate tax	
Calculate shipping	
Calculate total	

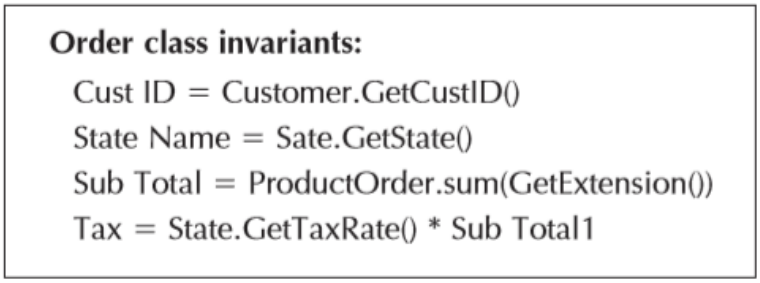
(a)



Gambar 8-20 Invarian pada Diagram Kelas

Sebuah kontrak harus berisi informasi yang diperlukan bagi seorang programmer untuk memahami apa yang harus dilakukan suatu metode (yaitu, mereka bersifat deklaratif). Informasi tersebut meliputi nama metode, nama kelas, nomor ID, objek klien, Use-Case terkait, deskripsi, argumen yang diterima, jenis data yang dikembalikan, dan kondisi pra dan pasca. Kontrak tidak memiliki deskripsi algoritmik terperinci tentang cara kerja metode ini. Deskripsi algoritmik terperinci biasanya didokumentasikan dalam spesifikasi metode (seperti yang dijelaskan nanti dalam bab ini). Dengan kata lain, kontrak terdiri dari informasi yang diperlukan untuk pengembang objek klien untuk mengetahui pesan apa yang dapat dikirim ke objek server dan apa yang dapat diharapkan klien sebagai balasannya. Gambar 8-22 menunjukkan contoh format kontrak.

Karena setiap kontrak dikaitkan dengan metode dan kelas tertentu, kontrak harus mendokumentasikannya. Nomor ID kontrak digunakan untuk memberikan pengidentifikasi unik untuk setiap kontrak. Elemen Klien (Konsumen) dari kontrak adalah daftar kelas dan metode yang mengirim pesan ke metode khusus ini. Daftar ini ditentukan dengan meninjau diagram urutan yang terkait dengan kelas server. Elemen Associated Use Cases adalah daftar use case dimana metode ini digunakan untuk merealisasikan implementasi dari use case tersebut. Use-Case yang tercantum di sini dapat ditemukan dengan meninjau kartu CRC kelas server dan diagram urutan terkait.



Gambar 8-21 Invarian dalam File Teks

Method Name:	Class Name:	ID:
Clients (Consumers):		
Associated Use Cases:		
Description of Responsibilities:		
Arguments Received:		
Type of Value Returned:		
Pre-Conditions:		
Post-Conditions:		

Gambar 8-22 Contoh Formulir Kontrak

Deskripsi Tanggung Jawab memberikan deskripsi informal tentang metode apa yang harus dilakukan, bukan bagaimana melakukannya. Argumen yang diterima adalah tipe data dari parameter yang diteruskan ke metode, dan nilai yang dikembalikan adalah tipe data dari nilai yang dikembalikan metode ke kliennya. Bersama dengan nama metode, mereka membentuk tanda tangan metode.

Elemen prakondisi dan pascakondisi adalah tempat prakondisi dan pascakondisi untuk metode dicatat. Ingatlah bahwa pra dan pascakondisi dapat ditulis dalam bahasa alami, bahasa semiformal, atau bahasa formal. Seperti halnya invarian, kami menyarankan Anda menggunakan Bahasa Batasan Objek UML.

Contoh Dalam contoh ini, kita kembali ke contoh urutan yang ditunjukkan pada Gambar 8-15, 8-19, 8-20, dan 8-21. Dalam hal ini, kami membatasi pembahasan pada desain metode addOrder untuk kelas Pelanggan. Keputusan pertama yang harus kita buat adalah bagaimana menentukan desain hubungan dari Pelanggan ke Pesanan. Dengan meninjau Gambar 8-15, 8-19, dan 8-20, kita melihat bahwa hubungan tersebut memiliki multiplisitas 0..* yang berarti bahwa sebuah instance pelanggan mungkin ada tanpa memiliki pesanan atau sebuah instance pelanggan dapat memiliki banyak perintah. Seperti yang ditunjukkan pada Gambar 8-15c, hubungan telah diubah menjadi atribut yang dapat berisi banyak instance dari kelas Order.

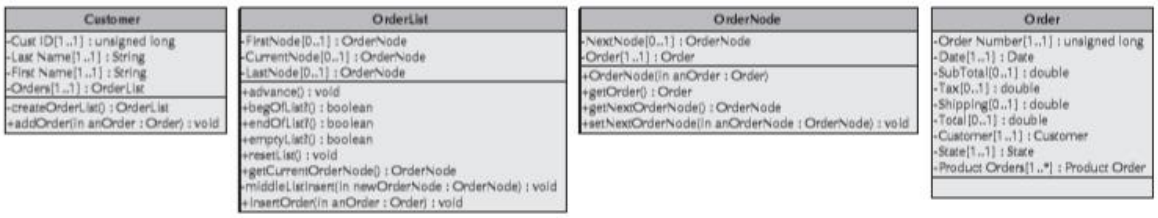
Namun, pertanyaan penting yang biasanya tidak muncul selama analisis adalah apakah objek urutan harus disimpan dalam urutan terurut atau tidak. Pertanyaan lain yang perlu dijawab untuk tujuan desain adalah berapa banyak pesanan yang dapat diharapkan oleh

pelanggan. Jawaban atas dua pertanyaan ini akan menentukan bagaimana kita harus mengatur pesanan dari perspektif objek pelanggan. Jika jumlah pesanan akan relatif kecil dan pesanan tidak harus disimpan dalam urutan yang diurutkan, maka menggunakan konstruksi bahasa pemrograman bawaan seperti vektor sudah cukup. Namun, jika jumlah pesanan akan menjadi besar atau pesanan harus disimpan dalam urutan yang diurutkan, maka beberapa bentuk struktur data yang diurutkan, seperti daftar tertaut, diperlukan. Sebagai contoh, kami berasumsi bahwa pesanan pelanggan perlu disimpan dalam urutan yang diurutkan dan jumlahnya akan banyak. Oleh karena itu, alih-alih menggunakan vektor untuk memuat pesanan, kami menggunakan daftar tertaut tunggal yang diurutkan.

Untuk menjaga desain kelas Pelanggan sedekat mungkin dengan representasi domain masalah, desain kelas Pelanggan didasarkan pada pola Iterator pada Gambar 8-13. Untuk tujuan sederhana, kami berasumsi bahwa pesanan dibuat sebelum dikaitkan dengan pelanggan tertentu. Jika tidak, mengingat batasan tambahan dari instance kelas State dan instance dari kelas Product Order yang ada sebelum instance Order dapat dibuat juga harus dipertimbangkan. Asumsi ini memungkinkan kita untuk mengabaikan fakta bahwa sebuah instance Negara dapat memiliki banyak pesanan, sebuah instance dari Order dapat memiliki banyak instance dari Product Order yang terkait dengannya, dan sebuah instance dari Product dapat memiliki banyak instance dari Product Order yang terkait dengannya, yang mengharuskan kita untuk merancang banyak wadah tambahan (vektor atau struktur data lainnya).

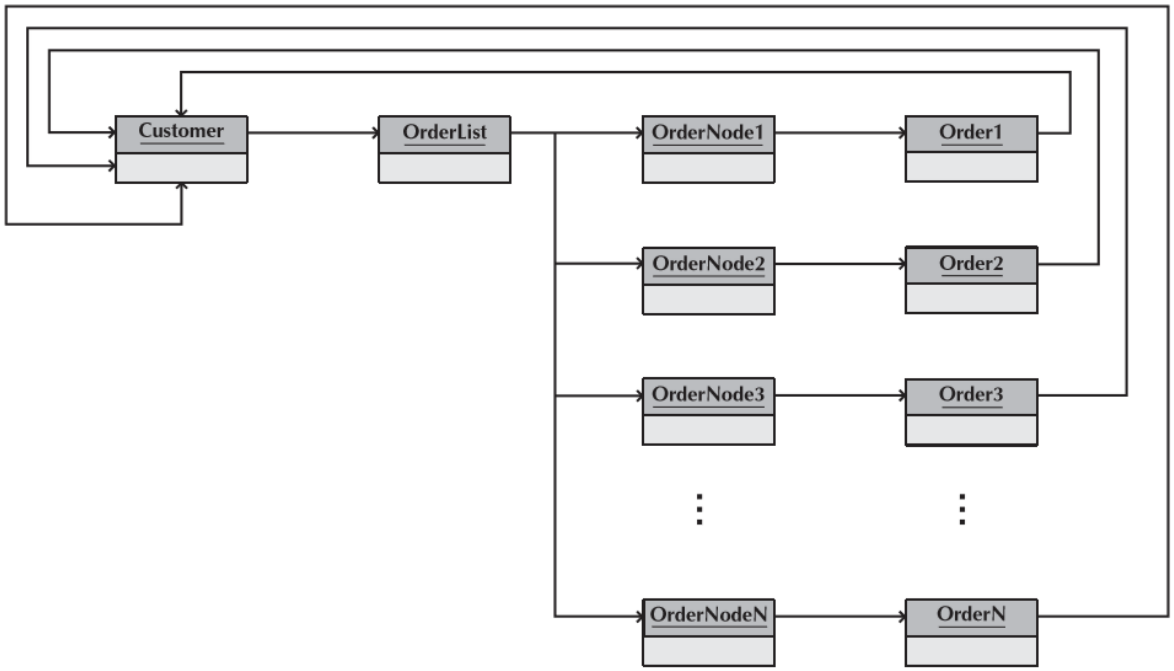
Berdasarkan semua hal di atas, sebuah fragmen diagram kelas baru dibuat yang mewakili hubungan berbasis daftar tertaut antara instance kelas Pelanggan dan instance kelas Order (lihat Gambar 8-23). Dengan hati-hati membandingkan Gambar 8-15 dan 8-23, kita melihat bahwa ide pola Iterator telah dimasukkan antara kelas Pelanggan dan Pesanan. Domain atribut berbasis hubungan Pesanan dari kelas Pelanggan telah diganti dengan DaftarPesanan untuk menunjukkan bahwa daftar pesanan akan dimuat dalam struktur data daftar. Gambar 8-24 menggambarkan representasi berbasis diagram objek tentang bagaimana hubungan antara instance pelanggan dan satu set instance pesanan disimpan dalam struktur data daftar tertaut tunggal yang diurutkan. Dalam kasus ini, kita melihat bahwa objek Customer memiliki objek OrderList yang terkait dengannya, setiap objek OrderList dapat memiliki N objek OrderNode, dan setiap objek OrderNode akan memiliki objek Order. Kami melihat bahwa setiap objek Pesanan dikaitkan dengan satu objek Pelanggan. Dengan membandingkan Gambar 8-15 dan 8-24, kita melihat bahwa maksud dari kendala multiplisitas atribut Pesanan Pelanggan, di mana pelanggan dapat memiliki banyak pesanan, dan kendala multiplisitas atribut Pelanggan Pesanan dimodelkan dengan benar. Terakhir, perhatikan bahwa salah satu operasi yang terdapat dalam kelas OrderList adalah metode pribadi. Kami akan kembali ke titik spesifik ini di bagian berikutnya yang membahas spesifikasi metode.

Menggunakan Gambar 8-22, 8-23, dan 8-24, kontrak untuk metode addOrder kelas Pelanggan dan metode insertOrder untuk kelas OrderList dapat ditentukan (lihat Gambar 8-25). Dalam kasus metode addOrder dari kelas Pelanggan, kita melihat bahwa hanya contoh kelas Pesanan yang menggunakan metode tersebut (lihat bagian Klien), bahwa metode tersebut hanya mengimplementasikan sebagian dari logika yang mendukung Use-Case addCustomerOrder (lihat Use-Case Terkait bagian), dan bahwa kontrak mencakup deskripsi singkat tentang tanggung jawab metode. Kita juga melihat bahwa metode menerima argumen tunggal bertipe Order dan tidak mengembalikan apa pun (void). Akhirnya, kita melihat bahwa baik prekondisi maupun postkondisi telah ditentukan. Prasyarat hanya menyatakan bahwa objek Pesanan baru tidak dapat dimasukkan dalam daftar Pesanan saat ini; yaitu, pesanan sebelumnya tidak dapat dikaitkan dengan pelanggan ini. Postcondition, di sisi lain, menetapkan bahwa daftar pesanan baru harus sama dengan daftar pesanan lama (@pre) ditambah objek pesanan baru (termasuk).



Gambar 8-23 Class Diagram Fragmen Customer to Order Relationship yang Dimodelkan sebagai Sorted Single Linked List.

Kontrak untuk metode insertOrder untuk kelas OrderList agak lebih sederhana daripada kontrak metode addOrder. Dari perspektif praktis, metode insertOrder mengimplementasikan bagian dari logika metode addOrder. Secara khusus, ini mengimplementasikan penyisipan aktual objek pesanan baru ke dalam struktur data spesifik yang dipilih untuk mengelola daftar objek Pesanan yang terkait dengan objek Pelanggan tertentu. Akibatnya, karena kita telah menentukan prakondisi dan pascakondisi untuk metode addOrder, kita tidak perlu lagi menentukan batasan yang sama untuk metode insertOrder. Namun, ini secara implisit meningkatkan ketergantungan objek Pelanggan pada implementasi yang dipilih untuk daftar pesanan pelanggan. Ini adalah contoh yang baik untuk berpindah dari domain masalah ke domain solusi. Sementara kami berfokus pada domain masalah selama analisis, implementasi sebenarnya dari daftar pesanan tidak pernah dipertimbangkan. Namun, karena kami sekarang sedang merancang implementasi hubungan antara objek Pelanggan dan objek Pesanan, kami harus menjauh dari bahasa pengguna akhir dan menuju bahasa pemrogram. Selama desain, fokus bergerak menuju pengoptimalan kode agar berjalan lebih cepat di komputer dan tidak mengkhawatirkan kemampuan pengguna akhir untuk memahami cara kerja bagian dalam sistem; dari sudut pandang pengguna akhir, sistem harus menjadi lebih dari kotak hitam tempat mereka berinteraksi. Saat kita melangkah lebih jauh ke dalam desain rinci implementasi kelas domain masalah, beberapa kelas domain solusi, seperti pendekatan untuk mengimplementasikan hubungan, akan masuk ke dalam spesifikasi lapisan domain masalah. Dalam contoh khusus ini, kelas OrderList dan OrderNode juga dapat digunakan untuk mengimplementasikan hubungan dari objek State ke objek Order, dari Objek Order ke objek Product Order, dan dari objek Product ke objek Product Order (lihat Gambar 8 -15). Mengingat contoh sederhana kami, orang dapat dengan jelas melihat bahwa menentukan desain lapisan domain masalah dapat mencakup banyak kelas domain solusi tambahan yang akan ditentukan pada lapisan domain masalah.



Gambar 8-24 Diagram Obyek Hubungan Pelanggan dengan Pesanan yang Dimodelkan sebagai Sorted Single Linked List

Method Name: addOrder	**Class Name:** Customer	**ID:** 36
Clients (consumers): Order		
Associated Use Cases: addCustomerOrder		
Description of Responsibilities: Terapkan perilaku yang diperlukan untuk menambahkan pesanan baru ke pelanggan yang sudah ada dengan menyimpan pesanan dalam urutan yang diurutkan berdasarkan nomor pesanan.		
Arguments Received: anOrder:Order		
Type of Value Returned: void		
Pre-Conditions: not orders.includes(anOrder)		
Post-Conditions: Orders = Orders@pre.including(anOrder)		

Method Name: insertOrder	Class Name: OrderList	ID: 123
Clients (consumers): Customer		
Associated Use Cases: addCustomerOrder		
Description of Responsibilities: Implementasikan penyisipan objek Order ke objek OrderNode dan kelola penyisipan objek OrderNode ke lokasi saat ini dalam daftar pesanan tertaut tunggal yang diurutkan.		
Arguments Received: anOrder:Order		
Type of Value Returned: void		
Pre-Conditions: None.		
Post-Conditions: None.		

Gambar 8-25 Contoh Kontrak untuk Metode addOrder dari Kelas Pelanggan dan Metode insertOrder dari Kelas OrderList

SPESIFIKASI METODE

Setelah analis mengomunikasikan gambaran besar tentang bagaimana sistem perlu disatukan, dia perlu menggambarkan kelas dan metode individual dengan cukup detail sehingga pemrogram dapat mengambil alih dan mulai menulis kode. Metode pada kartu CRC, diagram kelas, dan kontrak dijelaskan menggunakan spesifikasi metode. Spesifikasi metode adalah dokumen tertulis yang menyertakan instruksi eksplisit tentang cara menulis kode untuk mengimplementasikan metode. Biasanya, anggota tim proyek menulis spesifikasi untuk setiap metode dan kemudian meneruskannya ke pemrogram yang menulis kode selama implementasi proyek. Spesifikasi harus sangat jelas dan mudah dipahami, atau pemrogram akan diperlambat saat mencoba menguraikan instruksi yang tidak jelas atau tidak lengkap.

Tidak ada sintaks formal untuk spesifikasi metode, jadi setiap organisasi menggunakan formatnya sendiri, sering kali menggunakan formulir seperti pada Gambar 8-26. Formulir spesifikasi metode tipikal berisi empat komponen yang menyampaikan informasi yang dibutuhkan programmer untuk menulis kode yang sesuai: informasi umum, kejadian, penyampaian pesan, dan spesifikasi algoritma.

Informasi Umum

Bagian atas formulir pada Gambar 8-26 berisi informasi umum, seperti nama metode, nama kelas di mana implementasi metode ini akan berada, nomor ID, ID Kontrak (yang mengidentifikasi kontrak yang terkait dengan metode ini implementasi), programmer

ditugaskan, tanggal jatuh tempo, dan bahasa pemrograman target. Informasi ini digunakan untuk membantu mengelola upaya pemrograman.

Method Name:	Class Name:	ID:
Contract ID:	Programmer:	Date Due:
Programming Language: ❑ Visual Basic ❑ Smalltalk ❑ C++ ❑ Java		
Triggers/Events:		

Arguments Received: Data Type:	Notes:

Messages Sent & Arguments Passed: ClassName.MethodName:	Data Type:	Notes:

Arguments Returned: Data Type:	Notes:

Algorithm Specification:

Misc. Notes:

Gambar 8.26 Formulir Spesifikasi Metode

Event

Bagian kedua dari formulir digunakan untuk membuat daftar peristiwa yang memicu metode. Peristiwa adalah sesuatu yang terjadi atau terjadi. Mengklik mouse menghasilkan peristiwa mouse, menekan tombol menghasilkan peristiwa penekanan tombol—sebenarnya, hampir semua yang dilakukan pengguna menghasilkan peristiwa.

Di masa lalu, programmer menggunakan bahasa pemrograman prosedural yang berisi instruksi yang diimplementasikan dalam urutan yang telah ditentukan, sebagaimana ditentukan oleh sistem komputer, dan pengguna tidak diizinkan untuk menyimpang dari urutan. Banyak program saat ini yang digerakkan oleh peristiwa (misalnya, program yang ditulis dalam bahasa seperti Visual Basic, Objective C, C++, atau Java), dan program yang digerakkan oleh peristiwa mencakup metode yang dijalankan sebagai respons terhadap peristiwa yang diprakarsai oleh pengguna, sistem, atau metode lain. Setelah inisialisasi, sistem menunggu peristiwa terjadi. Ketika itu terjadi, sebuah metode dijalankan yang melakukan tugas yang sesuai, dan kemudian sistem menunggu sekali lagi.

Kami telah menemukan bahwa banyak pemrogram masih menggunakan spesifikasi metode saat memprogram dalam bahasa yang digerakkan oleh peristiwa, dan mereka menyertakan bagian peristiwa pada formulir untuk menangkap kapan metode akan dipanggil. Pemrogram lain telah beralih ke alat desain lain yang menangkap instruksi pemrograman yang digerakkan oleh peristiwa, seperti mesin status perilaku yang dijelaskan dalam Bab 6.

Message Passing

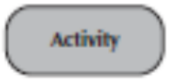
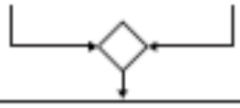
Bagian selanjutnya dari spesifikasi metode menjelaskan pesan yang lewat ke dan dari metode, yang diidentifikasi pada diagram urutan dan kolaborasi. Pemrogram perlu memahami argumen apa yang diteruskan, diteruskan, dan dikembalikan oleh metode karena argumen pada akhirnya diterjemahkan ke dalam atribut dan struktur data dalam metode yang sebenarnya.

Common Statements	Example
Action Statement	Profits = Revenues – Expenses Generate Inventory-Report
If Statement	IF Customer Not in the Customer Object Store THEN Add Customer record to Customer Object Store ELSE Add Current-Sale to Customer's Total-Sales Update Customer record in Customer Object Store
For Statement	FOR all Customers in Customer Object Store DO Generate a new line in the Customer-Report Add Customer's Total-Sales to Report-Total
Case Statement	CASE IF Income < 10,000: Marginal-tax-rate = 10 percent IF Income < 20,000: Marginal-tax-rate = 20 percent IF Income < 30,000: Marginal-tax-rate = 31 percent IF Income < 40,000: Marginal-tax-rate = 35 percent ELSE Marginal-Tax-Rate = 38 percent ENDCASE

Gambar 8-27 Bahasa Inggris Terstruktur

Spesifikasi Algoritma

Spesifikasi algoritma dapat ditulis dalam Bahasa Inggris Terstruktur atau beberapa jenis bahasa formal. Bahasa Inggris Terstruktur hanyalah cara formal menulis instruksi yang menggambarkan langkah-langkah suatu proses. Karena ini adalah langkah pertama menuju implementasi metode, ini terlihat seperti bahasa pemrograman sederhana. Bahasa Inggris Terstruktur menggunakan kalimat pendek yang dengan jelas menggambarkan pekerjaan apa yang dilakukan pada data apa. Ada banyak versi Bahasa Inggris Terstruktur karena tidak ada standar formal; setiap organisasi memiliki jenis Bahasa Inggris Terstrukturnya sendiri. Gambar 8-27 menunjukkan beberapa contoh pernyataan Bahasa Inggris Terstruktur yang umum digunakan.

Action: ■ Sederhananya, bagian dari perilaku yang tidak dapat dikomposisi. ■ Diberi label dengan namanya.	
Activity: ■ Digunakan untuk mewakili serangkaian tindakan. ■ Diberi label dengan namanya.	
Simpul Objek ■ Digunakan untuk merepresentasikan sebuah objek yang terhubung dengan sekumpulan objek mengalir. ■ Diberi label dengan nama kelasnya.	
Aliran kontrol: ■ Menunjukkan urutan eksekusi.	
Aliran objek: ■ Menunjukkan aliran suatu objek dari satu aktivitas (atau tindakan) ke aktivitas (atau tindakan) lain.	
Simpul awal: ■ Menggambarkan awal dari serangkaian tindakan atau kegiatan.	
Simpul akhir aktivitas: ■ Digunakan untuk menghentikan semua aliran kontrol dan aliran objek dalam suatu aktivitas (atau tindakan).	
Simpul akhir aliran: ■ Digunakan untuk menghentikan aliran kontrol tertentu atau aliran objek.	
Simpul keputusan: ■ Digunakan untuk mewakili kondisi pengujian untuk memastikan bahwa aliran kontrol atau aliran objek hanya turun satu jalur. ■ Dilabeli dengan kriteria keputusan untuk melanjutkan ke jalur tertentu.	
Simpul gabungan: ■ Digunakan untuk menyatukan kembali jalur keputusan yang berbeda yang dibuat menggunakan simpul keputusan.	
Simpul cabang: ■ Digunakan untuk membagi perilaku menjadi serangkaian aktivitas (atau tindakan) paralel atau bersamaan.	
Simpul berhubungan: ■ Digunakan untuk menyatukan kembali serangkaian aktivitas (atau tindakan) paralel atau bersamaan.	
Jalur renang: ■ Digunakan untuk memecah diagram aktivitas menjadi baris dan kolom untuk menetapkan aktivitas individu (atau tindakan) kepada individu atau objek yang bertanggung jawab untuk melaksanakan aktivitas (atau tindakan). ■ Diberi label dengan nama individu atau objek yang bertanggung jawab.	

Gambar 8-28 Sintaks untuk Diagram Aktivitas (Gambar 4-7)

Pernyataan tindakan adalah pernyataan sederhana yang melakukan beberapa tindakan. Pernyataan If mengontrol tindakan yang dilakukan di bawah kondisi yang berbeda, dan pernyataan For (atau pernyataan While) melakukan beberapa tindakan hingga beberapa kondisi tercapai. Pernyataan Kasus adalah bentuk lanjutan dari pernyataan If yang memiliki beberapa cabang yang saling eksklusif.

Jika algoritma suatu metode kompleks, alat yang dapat berguna untuk spesifikasi algoritma adalah diagram aktivitas UML (lihat Gambar 8-28 dan Bab 4). Ingat bahwa diagram aktivitas dapat digunakan untuk menentukan semua jenis proses. Jelas, spesifikasi algoritma mewakili suatu proses. Namun, karena sifat orientasi objek, proses cenderung sangat terdistribusi pada banyak metode kecil pada banyak objek. Perlu menggunakan diagram aktivitas untuk menentukan algoritme suatu metode, pada kenyataannya, dapat mengisyaratkan masalah dalam desain. Misalnya, metode harus didekomposisi lebih lanjut atau mungkin ada kelas yang hilang.

Bagian terakhir dari spesifikasi metode menyediakan ruang untuk informasi lain yang perlu dikomunikasikan kepada pemrogram, seperti perhitungan, aturan bisnis khusus, panggilan ke subrutin atau pustaka, dan masalah relevan lainnya. Ini juga dapat menunjukkan perubahan atau perbaikan yang akan dilakukan pada salah satu dokumentasi desain lainnya berdasarkan masalah yang dideteksi analis selama proses spesifikasi.

Contoh

Contoh ini melanjutkan penambahan pesanan baru untuk pelanggan yang dijelaskan di bagian sebelumnya (lihat Gambar 8-29). Meskipun dalam banyak kasus, karena ada perpustakaan kelas struktur data yang tersedia yang dapat Anda gunakan kembali dan oleh karena itu tidak perlu menentukan algoritme untuk dimasukkan ke dalam daftar tertaut tunggal yang diurutkan, kami menggunakannya sebagai contoh bagaimana spesifikasi metode dapat ulung. Bagian informasi umum dari spesifikasi mendokumentasikan nama metode, kelasnya, nomor ID uniknya, nomor ID dari kontrak terkait, programmer yang ditugaskan, tanggal pelaksanaannya, dan bahasa pemrograman yang akan digunakan. Kedua, pemicu/peristiwa yang menyebabkan metode ini dieksekusi diidentifikasi. Ketiga, tipe data argumen yang diteruskan ke metode ini didokumentasikan (Order). Keempat, karena kerumitan keseluruhan penyisipan node baru ke dalam daftar, kami telah memfaktorkan satu aspek spesifik dari algoritme ke dalam metode pribadi yang terpisah (middleListInsert()) dan kami telah menetapkan bahwa metode ini akan mengirim pesan ke instance dari kelas OrderNode dan kelas Order. Kelima, kita tentukan jenis nilai kembalian yang akan dihasilkan insertOrder. Dalam hal ini, metode insertOrder tidak akan mengembalikan apa pun (void). Akhirnya, kami menentukan algoritma yang sebenarnya. Dalam contoh ini, demi kelengkapan, kami menyediakan baik berbasis Bahasa Inggris Terstruktur (lihat Gambar 8-30) dan spesifikasi algoritma berbasis diagram aktivitas (lihat Gambar 8-31). Sebelumnya, kami menyatakan bahwa kami telah memperhitungkan logika penyisipan ke tengah daftar ke dalam metode pribadi yang terpisah: middleList Insert(). Gambar 8-32 menunjukkan logika metode ini. Bayangkan meruntuhkan logika ini kembali ke logika metode insertOrder, yaitu, mengganti aktivitas middleListInsert(newOrderNode) pada Gambar 8-31 dengan isi Gambar 8-32. Jelas, metode insertOrder akan lebih kompleks.

Method Name : insertOrder	**Class Name**: OrdrerList	**ID** : 100
Contract ID : 123	**Programmer** : Agus	**Date Due** : 1/1/2021

Programming Language: o Visual Basic o Smalltalk o C++ o Java		
Triggers/Events: Pelanggan melakukan pemesanan		
Arguments Received: **Data Types :**	**Notes :**	
Order	Order baru pelanggan baru	
Messages Sent & Arguments Passed: **ClassName.MethodName:**	**Data Type :**	**Notes:**
OrderNode.new()	Order	
OrderNode.getOrder()		
Order.getOrderNumber()		
OrderNode.setNextNode()	OrderNode	
self.middleListInsert()	OrderNode	
Arguments Received: **Data Types :**	**Notes :**	
Void		
Algorithm Specification: Lihat gambar 8-30 dan 8-31		
Misc. Notes: None		

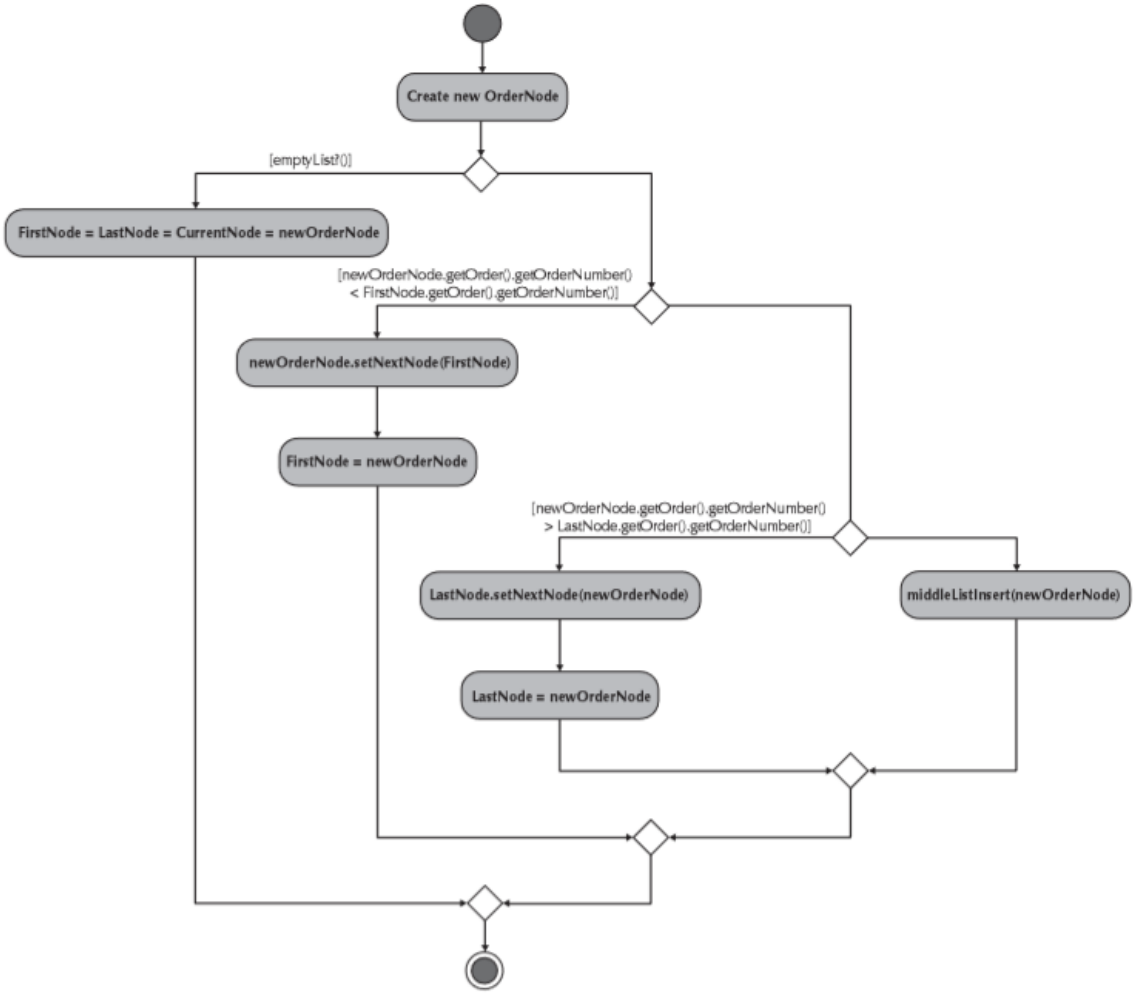
Gambar 8-29 Spesifikasi Metode untuk Metode insertOrder

```
Create new OrderNode with the new Order
IF emptyList?()
   FirstNode = LastNode = CurrentNode = newOrderNode
ELSE IF newOrderNode.getOrder().getOrderNumber() < FirstNode.getOrder().getOrderNumber()
   newOrderNode.setNextNode(FirstNode)
   FirstNode = newOrderNode
ELSE IF newOrderNode.getOrder().getOrderNumber() > LastNode.getOrder().getOrderNumber()
   LastNode.setNextNode(newOrderNode)
   LastNode = newOrderNode
ELSE
   middleListInsert(newOrderNode)
```

Gambar 8-30 Spesifikasi Algoritma Berbasis Bahasa Inggris Terstruktur untuk Metode insertOrder.

8.7 VERIFIKASI DAN VALIDASI DESAIN KELAS DAN METODE

Seperti semua model domain masalah sebelumnya, batasan, kontrak, dan spesifikasi metode perlu diverifikasi dan divalidasi. Mengingat bahwa kita terutama berurusan dengan domain masalah dalam bab ini, kendala dan kontrak diturunkan dari persyaratan fungsional dan representasi domain masalah. Namun, mereka berlaku untuk lapisan lain. Dalam hal ini, mereka akan diturunkan dari representasi domain solusi yang terkait dengan manajemen data (Bab 9), interaksi manusia-komputer (Bab 10), dan arsitektur sistem (Bab 11) lapisan. Mengingat semua masalah yang dijelaskan sebelumnya dengan kriteria desain (coupling, kohesi, dan connascence), spesifikasi tambahan, peluang penggunaan kembali, restrukturisasi dan optimalisasi desain, dan pemetaan ke bahasa implementasi, kemungkinan banyak modifikasi telah terjadi pada representasi analisis. dari domain masalah. Akibatnya, hampir semuanya harus diverifikasi ulang dan divalidasi ulang.

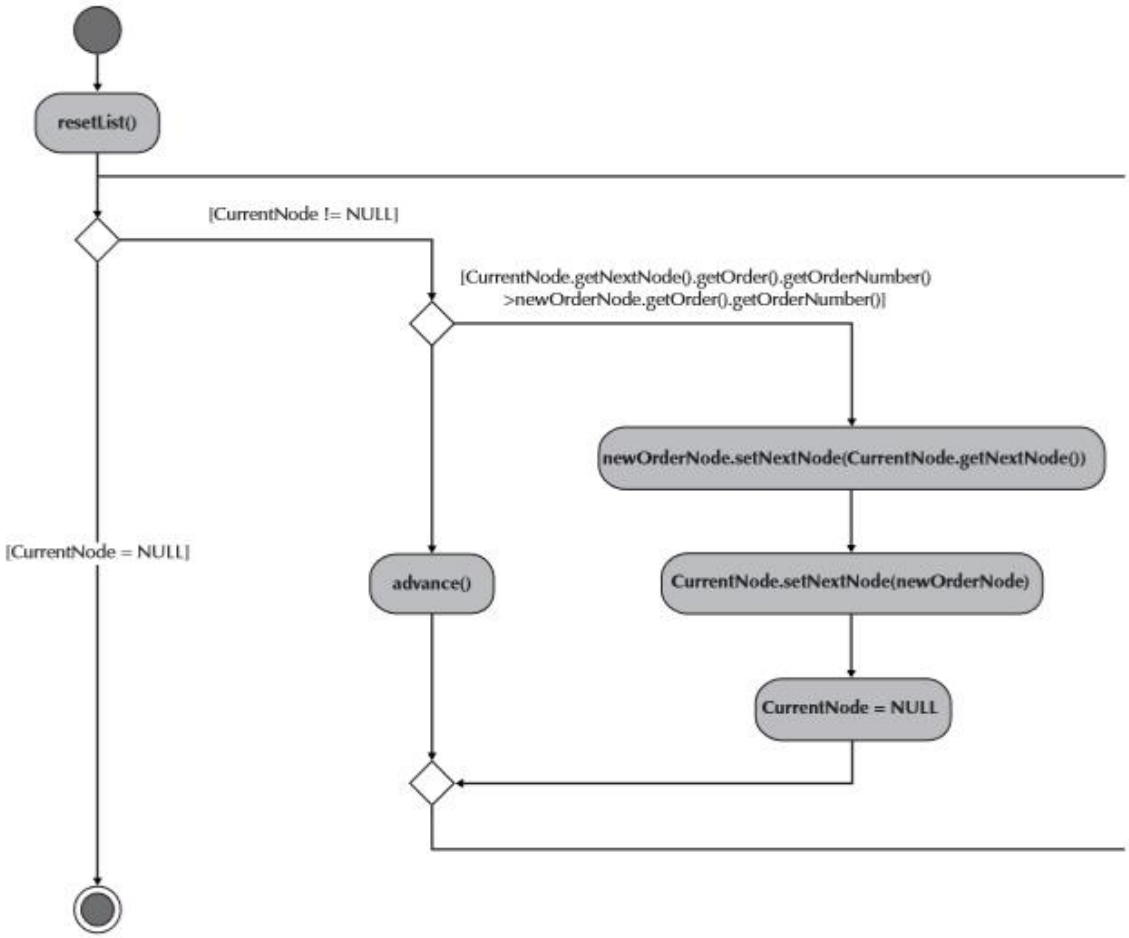


Gambar 8-31 Spesifikasi Algoritma Berbasis Activity Diagram untuk Metode Insertion Order

Pertama, kami merekomendasikan agar penelusuran semua representasi domain masalah yang berkembang dilakukan. Artinya, semua model fungsional (Bab 4) harus konsisten; semua model struktural (Bab 5) harus konsisten; semua model perilaku (Bab 6) harus konsisten; dan model fungsional, struktural, dan perilaku harus seimbang (Bab 7).

Kedua, semua batasan, kontrak, dan spesifikasi metode harus diuji. Cara terbaik untuk melakukannya adalah dengan memainkan peran sistem menggunakan skenario yang berbeda dari Use-Case. Dalam hal ini, kita harus menerapkan invarian pada kartu CRC yang dikembangkan (lihat Gambar 8-19), pra dan pascakondisi pada formulir kontrak (lihat Gambar 8-22 dan 8-25), dan desain setiap metode yang ditentukan dengan formulir spesifikasi metode (lihat Gambar 8-26 dan 8-29) dan spesifikasi algoritma (lihat Gambar 8-30, 8-31, dan 8-32).

Mengingat jumlah verifikasi dan validasi fidelitas semua model yang telah kami lakukan pada sistem yang berkembang, mungkin tampak seperti berlebihan untuk melakukan hal di atas lagi. Namun, mengingat volume murni perubahan yang dapat terjadi selama desain, sangat penting untuk menguji model secara menyeluruh lagi sebelum sistem diimplementasikan. Faktanya, pengujian sangat penting untuk pendekatan pengembangan tangkas, pengujian membentuk tulang punggung virtual dari metodologi tersebut. Tanpa pengujian menyeluruh, tidak ada jaminan bahwa sistem yang diterapkan akan mengatasi masalah yang sedang dipecahkan. Setelah sistem diimplementasikan, pengujian menjadi lebih penting (lihat Bab 12).



Gambar 8-32 Spesifikasi Algoritma Berbasis Activity Diagram untuk Metode middleListInsert

Pertanyaan

1. Apa karakteristik dasar dari sistem berorientasi objek?
2. Apa itu ikatan dinamis?
3. Definisi polimorfisme. Berikan satu contoh penggunaan polimorfisme yang baik dan satu contoh penggunaan polimorfisme yang buruk.
4. Apa itu konflik warisan? Bagaimana konflik pewarisan mempengaruhi desain?
5. Mengapa pembatalan metode merupakan hal yang buruk?
6. Berikan pedoman untuk menghindari masalah dengan konflik warisan.
7. Mengapa penting untuk mengetahui bahasa pemrograman berorientasi objek mana yang akan digunakan untuk mengimplementasikan sistem?
8. Jenis konflik pewarisan tambahan apa yang ada saat menggunakan pewarisan berganda?
9. Apa hukum Demeter?
10. Apa enam jenis coupling interaksi? Berikan satu contoh coupling interaksi yang baik dan satu contoh coupling interaksi yang buruk.
11. Apa tujuh jenis kohesi metode? Berikan satu contoh kohesi metode yang baik dan satu contoh kohesi metode yang buruk.
12. Apa empat jenis kohesi kelas? Berikan satu contoh dari masing-masing jenis.
13. Apa lima jenis hati nurani yang dijelaskan dalam teks Anda? Berikan satu contoh dari masing-masing jenis.
14. Saat merancang kelas tertentu, jenis spesifikasi tambahan apa untuk kelas yang mungkin diperlukan?

15. Apa itu pengecualian?
16. Apa itu kendala? Apa tiga jenis kendala yang berbeda?
17. Apa itu pola, kerangka kerja, perpustakaan kelas, dan komponen? Bagaimana mereka digunakan untuk meningkatkan desain sistem yang berkembang?
18. Bagaimana pemfaktoran dan normalisasi digunakan dalam merancang sistem objek?
19. Apa saja cara berbeda untuk mengoptimalkan sistem objek?
20. Apa kelemahan khas dari optimasi sistem?
21. Apa tujuan dari sebuah kontrak? Bagaimana kontrak digunakan?
22. Apa itu Bahasa Batasan Objek? Apa tujuannya?
23. Apa itu Bahasa Inggris Terstruktur? Apa tujuannya?
24. Apa itu invarian? Bagaimana invarian dimodelkan dalam desain kelas? Berikan contoh invarian untuk kelas karyawan per jam menggunakan Bahasa Batasan Objek.
25. Buat kontrak untuk metode pembayaran komputasi yang terkait dengan kelas karyawan per jam. Tentukan preconditions dan postconditions menggunakan Object Constraint Language.
26. Bagaimana Anda menentukan algoritma metode? Berikan contoh spesifikasi algoritme untuk metode pembayaran komputasi yang terkait dengan kelas karyawan per jam menggunakan Bahasa Inggris Terstruktur.
27. Bagaimana Anda menentukan algoritma metode? Berikan contoh spesifikasi algoritme untuk metode pembayaran komputasi yang terkait dengan kelas karyawan per jam menggunakan diagram aktivitas.
28. Bagaimana metode ditentukan? Berikan contoh spesifikasi metode untuk metode pembayaran komputasi yang terkait dengan kelas karyawan per jam.

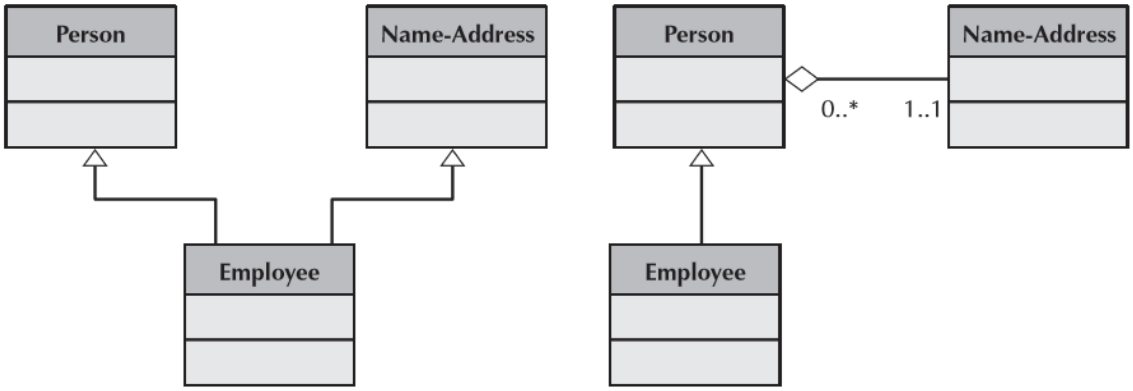
Latihan

A. Untuk masalah A Real Estate Inc. di Bab 4 (latihan I, J, dan K), 5 (latihan P dan Q), 6 (latihan D), dan 7 (latihan A):
    a. Pilih salah satu kelas dan buat satu set invarian untuk atribut dan hubungan dan tambahkan ke kartu CRC untuk kelas tersebut.
    b. Pilih salah satu metode di kelas yang Anda pilih dan buat kontrak dan spesifikasi metode untuknya. Gunakan OCL untuk menentukan pra atau pascakondisi apa pun dan gunakan Bahasa Inggris Terstruktur dan diagram aktivitas untuk menentukan algoritme.
B. Untuk masalah A Video Store di Bab 4 (latihan L, M, NK), 5 (latihan R dan S), 6 (latihan E), dan 7 (latihan B):
    a. Pilih salah satu kelas dan buat satu set invarian untuk atribut dan hubungan dan tambahkan ke kartu CRC untuk kelas tersebut.
    b. Pilih salah satu metode di kelas yang Anda pilih dan buat kontrak dan spesifikasi metode untuknya. Gunakan OCL untuk menentukan kondisi awal atau akhir dan gunakan Bahasa Inggris Terstruktur dan diagram aktivitas untuk menentukan algoritme.
C. Untuk masalah keanggotaan gym di Bab 4 (latihan O, P, dan Q), 5 (latihan T dan U), 6 (latihan F), dan 7 (latihan C):
    a. Pilih salah satu kelas dan buat satu set invarian untuk atribut dan hubungan dan tambahkan ke kartu CRC untuk kelas tersebut.
    b. Pilih salah satu metode di kelas yang Anda pilih dan buat kontrak dan spesifikasi metode untuknya. Gunakan OCL untuk menentukan pra atau pascakondisi apa pun dan gunakan Bahasa Inggris Terstruktur dan diagram aktivitas untuk menentukan algoritme.
D. Untuk soal Piknik R Us di Bab 4 (latihan R, S, dan T), 5 (latihan V dan W), 6 (latihan G), dan 7 (latihan D):

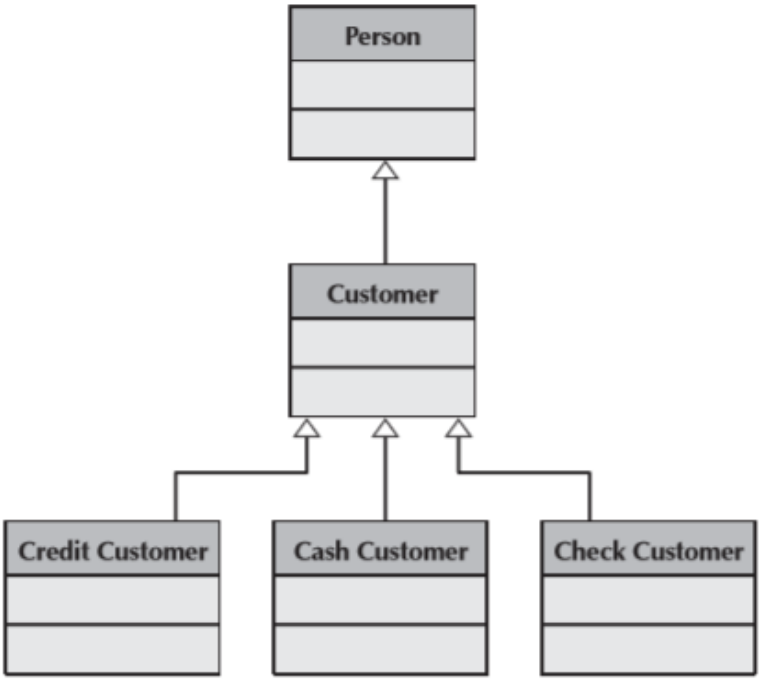
a. Pilih salah satu kelas dan buat satu set invarian untuk atribut dan hubungan dan tambahkan ke kartu CRC untuk kelas tersebut.
b. Pilih salah satu metode di kelas yang Anda pilih dan buat kontrak dan spesifikasi metode untuknya. Gunakan OCL untuk menentukan pra atau pascakondisi apa pun dan gunakan Bahasa Inggris Terstruktur dan diagram aktivitas untuk menentukan algoritme.

E. Untuk masalah Klub Bulanan di Bab 4 (latihan U, V, dan W), 5 (latihan X dan Y), 6 (latihan H), dan 7 (latihan E):
a. Pilih salah satu kelas dan buat satu set invarian untuk atribut dan hubungan dan tambahkan ke kartu CRC untuk kelas tersebut.
b. Pilih salah satu metode di kelas yang Anda pilih dan buat kontrak dan spesifikasi metode untuknya. Gunakan OCL untuk menentukan pra atau pascakondisi apa pun dan gunakan Bahasa Inggris Terstruktur dan diagram aktivitas untuk menentukan algoritme.

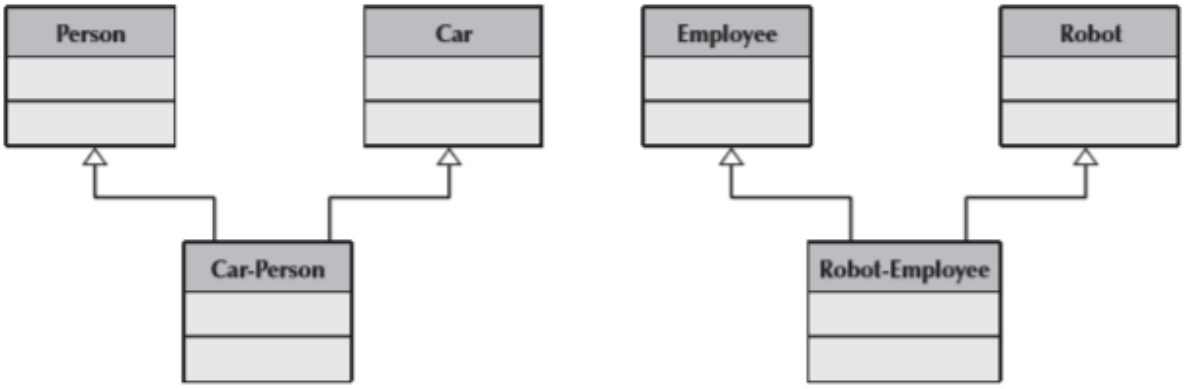
F. Jelaskan perbedaan makna antara dua diagram kelas berikut. Manakah model yang lebih baik? Mengapa?



G. Dari perspektif kohesi, coupling, dan connascence, apakah diagram kelas berikut merupakan model yang baik? Mengapa atau mengapa tidak?



H. Dari perspektif kohesi, coupling, dan connascence, apakah diagram kelas berikut merupakan model yang baik? Mengapa atau mengapa tidak?



I. Buat satu set konflik pewarisan untuk dua struktur pewarisan dalam diagram kelas latihan H.

Kasus Kecil

1. Bos Anda telah berkecimpung di bidang pengembangan perangkat lunak selama tiga puluh tahun. Dia selalu bangga dengan kemampuannya untuk mengadaptasi keterampilannya dari satu pendekatan untuk mengembangkan perangkat lunak ke pendekatan berikutnya. Misalnya, dia tidak memiliki masalah dalam mempelajari analisis dan desain terstruktur pada awal 1980-an dan rekayasa informasi pada awal 1990-an. Dia bahkan memahami keuntungan dari pengembangan aplikasi yang cepat. Tapi tempo hari, ketika Anda dan dia berbicara tentang keuntungan dari pendekatan berorientasi objek, dia menjadi benar-benar bingung. Dia berpikir bahwa karakteristik seperti polimorfisme dan pewarisan merupakan keuntungan untuk sistem berorientasi objek. Namun, ketika Anda menjelaskan masalah dengan konflik pewarisan, kemampuan pendefinisian ulang, dan kebutuhan akan konsistensi semantik di berbagai implementasi metode yang berbeda, dia siap untuk menyerah begitu saja. Lebih buruk lagi, Anda kemudian menjelaskan pentingnya kontrak dalam mengendalikan pengembangan sistem. Pada titik konservasi ini, dia pada dasarnya menyerah. Saat dia berjalan pergi, Anda mendengar dia mengatakan sesuatu seperti "Saya rasa itu benar, terlalu sulit untuk mengajari seekor anjing tua trik baru."

   Menjadi karyawan dan teman yang setia, Anda memutuskan untuk menulis tutorial singkat untuk memberi bos Anda tentang pengembangan sistem berorientasi objek. Sebagai langkah pertama, buat garis besar detail untuk tutorial. Sebagai contoh halus, gunakan kriteria desain yang baik, seperti coupling dan kohesi, dalam desain garis besar tutorial Anda.

2. Anda telah bekerja dengan masalah manajemen profesional dan ilmiah (PSSM) untuk waktu yang cukup lama. Anda harus kembali dan menyegarkan ingatan Anda tentang masalah tersebut sebelum mencoba menyelesaikan situasi ini. Lihat kembali solusi Anda untuk Minicase 3 di Bab 7.
   a. Untuk setiap kelas dalam model struktural, menggunakan OCL, buat satu set invarian untuk atribut dan hubungan dan tambahkan ke kartu CRC untuk kelas.
   b. Pilih salah satu kelas dalam model struktural. Buat kontrak untuk setiap metode di kelas itu. Pastikan untuk menggunakan OCL untuk menentukan prakondisi dan pascakondisi. Jadilah selengkap mungkin.

c. Buat spesifikasi metode untuk setiap metode di kelas yang Anda pilih untuk pertanyaan b. Gunakan bahasa Inggris Terstruktur dan diagram aktivitas untuk spesifikasi algoritme.

3. Anda telah bekerja dengan masalah Kendaraan Perjalanan Liburan cukup lama. Anda harus kembali dan menyegarkan ingatan Anda tentang masalah tersebut sebelum mencoba menyelesaikan situasi ini. Lihat kembali solusi Anda Minicase 4 di Bab 7.

   Dalam sistem baru untuk Kendaraan Perjalanan Liburan, pengguna sistem mengikuti proses dua tahap untuk mencatat informasi lengkap tentang semua kendaraan yang terjual. Ketika sebuah RV atau trailer pertama kali tiba di perusahaan dari pabrikan, petugas dari departemen persediaan membuat catatan kendaraan baru untuk itu dalam sistem komputer. Data yang dimasukkan saat ini meliputi informasi deskriptif dasar pada kendaraan seperti pabrikan, nama, model, tahun, biaya dasar, dan biaya pengiriman. Ketika kendaraan dijual, catatan kendaraan baru diperbarui untuk mencerminkan persyaratan penjualan akhir dan opsi yang dipasang dealer ditambahkan ke kendaraan. Informasi ini dimasukkan ke dalam sistem pada saat penjualan saat tenaga penjual melengkapi faktur penjualan.

   Ketika tiba saatnya petugas menyelesaikan pencatatan kendaraan baru, petugas memilih opsi menu dari sistem, yang disebut Finalisasi Catatan Kendaraan Baru. Tugas yang terlibat dalam proses ini dijelaskan di bawah ini.

   Ketika pengguna memilih Finalize New Vehicle Record dari menu sistem, pengguna segera diminta untuk memasukkan nomor seri kendaraan baru. Nomor seri ini digunakan untuk mengambil catatan kendaraan baru untuk kendaraan dari penyimpanan sistem. Jika catatan tidak dapat ditemukan, nomor seri mungkin tidak valid. Nomor seri kendaraan kemudian digunakan untuk mengambil catatan opsi yang menjelaskan opsi yang dipasang dealer yang ditambahkan ke kendaraan atas permintaan pelanggan. Mungkin ada nol atau lebih pilihan. Biaya opsi yang ditentukan pada catatan opsi dijumlahkan. Kemudian, biaya dealer dihitung menggunakan biaya dasar kendaraan, biaya pengiriman, dan biaya opsi total. Catatan kendaraan baru yang telah selesai diteruskan kembali ke modul panggilan.

   a. Perbarui model struktural (kartu CRC dan diagram kelas) dengan informasi tambahan ini.
   b. Untuk setiap kelas dalam model struktural, menggunakan OCL, buat satu set invarian untuk atribut dan hubungan dan tambahkan ke kartu CRC untuk kelas.
   c. Pilih salah satu kelas dalam model struktural. Buat kontrak untuk setiap metode di kelas itu. Pastikan untuk menggunakan OCL untuk menentukan prakondisi dan pascakondisi. Jadilah selengkap mungkin.
   d. Buat spesifikasi metode untuk setiap metode di kelas yang Anda pilih untuk pertanyaan b. Gunakan bahasa Inggris Terstruktur dan diagram aktivitas untuk spesifikasi algoritme

BAB 9

DESAIN LAPISAN MANAJEMEN DATA

Sebuah tim proyek merancang lapisan manajemen data dari suatu sistem menggunakan proses empat langkah: memilih format penyimpanan, memetakan kelas domain masalah ke format yang dipilih, mengoptimalkan penyimpanan untuk bekerja secara efisien, dan kemudian merancang akses data yang diperlukan dan kelas manipulasi. Bab ini menjelaskan berbagai cara objek dapat disimpan dan beberapa karakteristik penting yang harus dipertimbangkan saat memilih di antara format persistensi objek. Ini menjelaskan kelas domain masalah ke objek proses pemetaan format ketekunan untuk format persistensi objek yang paling penting. Karena format penyimpanan yang paling populer saat ini adalah database relasional, bab ini berfokus pada optimalisasi database relasional dari perspektif penyimpanan dan akses. Kami menjelaskan efek yang dimiliki persyaratan nonfungsional pada lapisan manajemen data. Bab ini akhirnya menjelaskan bagaimana merancang kelas akses dan manipulasi data dan menjelaskan bagaimana memastikan kesetiaan lapisan manajemen data.

9.1 Tujuan :

- Menjadi akrab dengan beberapa format persistensi objek.
- Mampu memetakan objek domain masalah ke format persistensi objek yang berbeda.
- Mampu menerapkan langkah-langkah normalisasi ke database relasional.
- Mampu mengoptimalkan database relasional untuk penyimpanan dan akses objek.
- Menjadi akrab dengan indeks untuk database relasional.
- Mampu memperkirakan ukuran database relasional.
- Memahami pengaruh persyaratan nonfungsional pada lapisan manajemen data.
- Mampu merancang kelas akses dan manipulasi data.

9.2 Pendahuluan

Aplikasi tidak banyak berguna tanpa data yang didukungnya. Seberapa berguna aplikasi multimedia yang tidak mendukung gambar atau suara? Mengapa seseorang masuk ke sistem untuk menemukan informasi jika waktu yang dibutuhkannya lebih sedikit untuk menemukan informasi secara manual? Salah satu keluhan utama oleh pengguna akhir adalah bahwa sistem final terlalu lambat, sehingga untuk menghindari keluhan seperti itu, anggota tim proyek harus memberikan waktu selama desain untuk memastikan dengan hati-hati bahwa file atau database bekerja secepat mungkin. Pada saat yang sama, tim harus menekan biaya perangkat keras dengan meminimalkan ruang penyimpanan yang dibutuhkan aplikasi. Tujuan memaksimalkan akses ke objek dan meminimalkan jumlah ruang yang diambil untuk menyimpan objek dapat bertentangan, dan merancang efisiensi persistensi objek biasanya memerlukan trade-off.

Desain lapisan manajemen data mengatasi masalah ini. Ini mencakup desain akses data dan kelas manipulasi dan penyimpanan data aktual. Desain kelas akses dan manipulasi data harus memastikan independensi kelas domain masalah dari format penyimpanan data. Dengan demikian, kelas akses dan manipulasi data menangani semua komunikasi dengan database. Dengan cara ini, domain masalah dipisahkan dari penyimpanan objek, memungkinkan penyimpanan objek diubah tanpa mempengaruhi kelas domain masalah.

Komponen penyimpanan data mengatur bagaimana data disimpan dan ditangani oleh program yang menjalankan sistem. Komponen penyimpanan data terdiri dari satu set kelas persistensi objek. Desain persistensi objek yang efektif mengurangi kemungkinan berakhir

dengan sistem yang tidak efisien, waktu respons sistem yang lama, dan pengguna yang tidak dapat memperoleh informasi yang mereka butuhkan dengan cara yang mereka butuhkan—semuanya dapat memengaruhi keberhasilan proyek. Dari perspektif praktis, ada lima tipe dasar format yang dapat digunakan untuk menyimpan objek untuk sistem aplikasi: file (berurutan dan acak), database berorientasi objek, database relasional objek, database relasional, atau penyimpanan data NoSQL. Setiap jenis memiliki karakteristik tertentu yang membuatnya lebih sesuai untuk beberapa jenis sistem daripada yang lain. Setelah format persistensi objek dipilih untuk mendukung sistem, objek domain masalah perlu mendorong desain penyimpanan objek yang sebenarnya. Kemudian penyimpanan objek perlu dirancang untuk mengoptimalkan efisiensi pemrosesannya

Order Number	Date	Cust ID	Last Name	First Name	Amount	Tax	Total	Prior Customer	Payment Type
234	11/23/00	2242	DeBerry	Ann	$ 90.00	$5.85	$ 95.85	Y	MC
235	11/23/00	9500	Chin	April	$ 12.00	$0.60	$ 12.60	Y	VISA
236	11/23/00	1556	Fracken	Chris	$ 50.00	$2.50	$ 52.50	N	VISA
237	11/23/00	2242	DeBerry	Ann	$ 75.00	$4.88	$ 79.88	Y	AMEX
238	11/23/00	2242	DeBerry	Ann	$ 60.00	$3.90	$ 63.90	Y	MC
239	11/23/00	1035	Black	John	$ 90.00	$4.50	$ 94.50	Y	AMEX
240	11/23/00	9501	Kaplan	Bruce	$ 50.00	$2.50	$ 52.50	N	VISA
241	11/23/00	1123	Williams	Mary	$120.00	$9.60	$129.60	N	MC
242	11/24/00	9500	Chin	April	$ 60.00	$3.00	$ 63.00	Y	VISA
243	11/24/00	4254	Bailey	Ryan	$ 90.00	$4.50	$ 94.50	Y	VISA
244	11/24/00	9500	Chin	April	$ 24.00	$1.20	$ 25.20	Y	VISA
245	11/24/00	2242	DeBerry	Ann	$ 12.00	$0.78	$ 12.78	Y	AMEX
246	11/24/00	4254	Bailey	Ryan	$ 20.00	$1.00	$ 21.00	Y	MC
247	11/24/00	2241	Jones	Chris	$ 50.00	$2.50	$ 52.50	N	VISA
248	11/24/00	4254	Bailey	Ryan	$ 12.00	$0.60	$ 12.60	Y	AMEX
249	11/24/00	5927	Lee	Diane	$ 50.00	$2.50	$ 52.50	N	AMEX
250	11/24/00	2242	DeBerry	Ann	$ 12.00	$0.78	$ 12.78	Y	MC
251	11/24/00	9500	Chin	April	$ 15.00	$0.75	$ 15.75	Y	MC
252	11/24/00	2242	DeBerry	Ann	$132.00	$8.58	$140.58	Y	MC
253	11/24/00	2242	DeBerry	Ann	$ 72.00	$4.68	$ 76.68	Y	AMEX

Gambar 9-1 File Pesanan Pelanggan

9.3 FORMAT PERSISTEN OBJEK

Masing-masing tipe persistensi objek dijelaskan di bagian ini. File adalah daftar data elektronik yang telah dioptimalkan untuk melakukan transaksi tertentu. Sebagai contoh, Gambar 9-1 menunjukkan file pesanan pelanggan dengan informasi tentang pesanan pelanggan, dalam bentuk yang digunakan, sehingga informasi tersebut dapat diakses dan diproses dengan cepat oleh sistem.

Database adalah kumpulan pengelompokan informasi, yang masing-masing terkait satu sama lain dalam beberapa cara (misalnya, melalui bidang umum). Pengelompokan informasi yang logis dapat mencakup kategori seperti data pelanggan, informasi tentang pesanan, informasi produk, dan sebagainya. Sistem manajemen database (DBMS) adalah perangkat lunak yang membuat dan memanipulasi database ini (lihat Gambar 9-2 untuk contoh database relasional). DBMS pengguna akhir seperti Microsoft Access mendukung database skala kecil yang digunakan untuk meningkatkan produktivitas pribadi, sedangkan DBMS perusahaan, seperti DB2, Versant, dan Oracle, dapat mengelola sejumlah besar data dan mendukung aplikasi yang menjalankan seluruh perusahaan. DBMS pengguna akhir secara signifikan lebih murah dan lebih mudah digunakan oleh pengguna pemula daripada rekan

perusahaannya, tetapi tidak memiliki fitur atau kemampuan yang diperlukan untuk mendukung misi penting atau sistem skala besar.

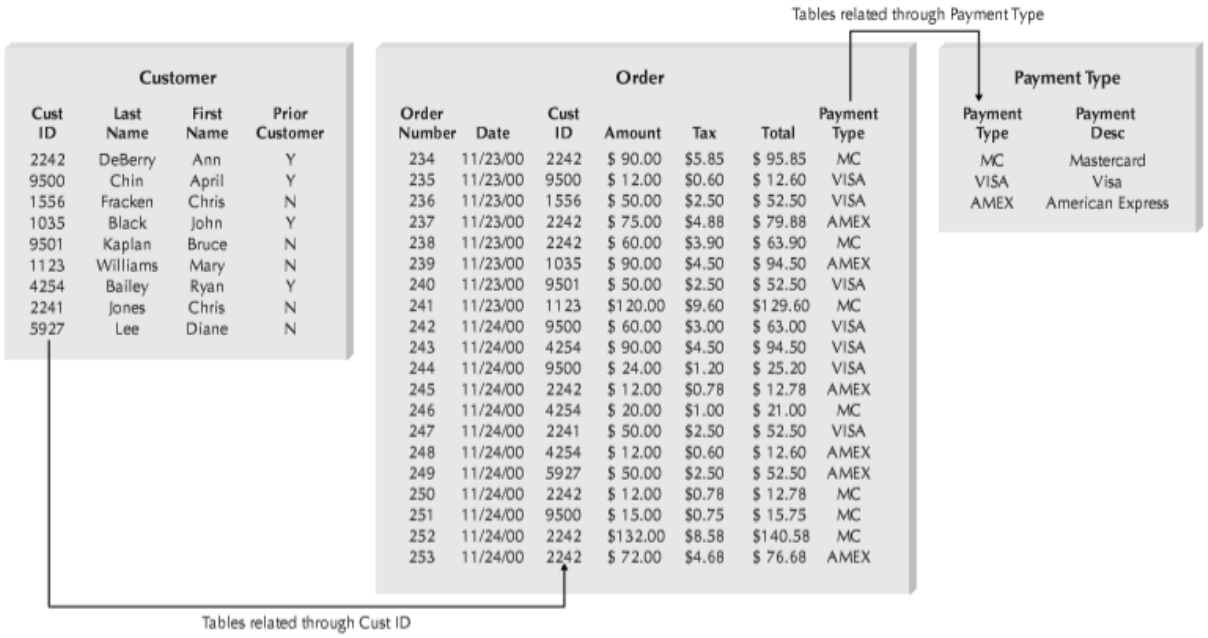
File Akses Berurutan dan Acak

Dari perspektif praktis, sebagian besar bahasa pemrograman berorientasi objek mendukung file akses sekuensial dan acak sebagai bagian dari bahasa. Di bagian ini, kami menjelaskan apa itu akses sekuensial dan file akses acak. Kami juga menjelaskan bagaimana akses sekuensial dan file akses acak digunakan untuk mendukung aplikasi. Misalnya, mereka dapat digunakan untuk mendukung file master, file pencarian, file transaksi, file audit, dan file riwayat.

File akses berurutan memungkinkan hanya operasi file berurutan yang akan dilakukan (misalnya, membaca, menulis, dan mencari). File akses sekuensial sangat efisien untuk operasi sekuensial yang memproses semua objek secara berurutan, seperti penulisan laporan. Namun, untuk operasi acak, seperti menemukan atau memperbarui objek tertentu, mereka sangat tidak efisien. Rata-rata, 50 persen konten file akses berurutan harus ditelusuri sebelum menemukan objek tertentu yang menarik dalam file. Mereka datang dalam dua rasa: dipesan dan tidak dipesan.

File akses sekuensial tidak berurutan pada dasarnya adalah daftar elektronik informasi yang disimpan di disk. File yang tidak diurutkan diatur secara serial (yaitu, urutan file adalah urutan objek yang ditulis ke file). Biasanya, objek baru hanya ditambahkan ke akhir file.File akses berurutan yang dipesan ditempatkan ke dalam urutan tertentu yang diurutkan (misalnya, dalam urutan menaik berdasarkan nomor pelanggan). Ada overhead yang terkait dengan menyimpan file dalam urutan tertentu yang diurutkan. Perancang file dapat menyimpan file dalam urutan yang diurutkan dengan selalu membuat file baru setiap kali terjadi penghapusan atau penambahan, atau dia dapat melacak urutan yang diurutkan melalui penggunaan pointer, yang merupakan informasi tentang lokasi file catatan terkait. Pointer ditempatkan di akhir setiap record, dan pointer tersebut “menunjuk” ke record berikutnya dalam rangkaian atau set. Struktur data/file yang mendasari dalam hal ini adalah struktur data daftar tertaut yang ditunjukkan pada bab sebelumnya.

File akses acak hanya mengizinkan operasi file acak atau langsung untuk dilakukan. Jenis file ini dioptimalkan untuk operasi acak, seperti menemukan dan memperbarui objek tertentu. File akses acak biasanya memiliki waktu respons yang lebih cepat untuk menemukan dan memperbarui operasi daripada jenis file lainnya. Namun, karena tidak mendukung pemrosesan sekuensial, aplikasi seperti penulisan laporan sangat tidak efisien. Berbagai metode untuk mengimplementasikan file akses acak berada di luar cakupan buku ini.



Gambar 9-2 Database Pesanan Pelanggan

Ada kalanya diperlukan untuk dapat memproses file baik secara berurutan maupun acak. Salah satu cara sederhana untuk melakukannya adalah dengan menggunakan file sekuensial yang berisi daftar utama (bidang di mana file akan disimpan dalam urutan yang diurutkan) dan file akses acak untuk objek yang sebenarnya. Ini meminimalkan biaya penambahan dan penghapusan ke file sekuensial sambil memungkinkan file acak diproses secara berurutan dengan hanya meneruskan utama ke file acak untuk mengambil setiap objek secara berurutan. Ini juga memungkinkan pemrosesan acak cepat terjadi dengan hanya menggunakan file akses acak, sehingga mengoptimalkan keseluruhan biaya pemrosesan file. Namun, jika file objek perlu diproses secara acak dan berurutan, pengembang harus mempertimbangkan untuk menggunakan database (relasional, relasional objek, atau berorientasi objek).

Ada banyak jenis file aplikasi yang berbeda—misalnya, file master, file pencarian, file transaksi, file audit, dan file riwayat. File master menyimpan informasi inti yang penting bagi bisnis dan, lebih khusus lagi, untuk aplikasi, seperti informasi pesanan atau informasi pengiriman pelanggan. Mereka biasanya disimpan untuk jangka waktu yang lama, dan catatan baru ditambahkan ke akhir file saat pesanan baru atau pelanggan baru ditangkap oleh sistem. Jika perubahan perlu dilakukan pada catatan yang ada, program harus ditulis untuk memperbarui informasi lama.

File pencarian berisi nilai statis, seperti daftar kode ZIP (kode pos) yang valid atau nama negara. Biasanya, daftar tersebut digunakan untuk validasi. Misalnya, jika alamat surat pelanggan dimasukkan ke dalam file master, nama negara divalidasi terhadap file pencarian yang berisi negara tersebut. untuk memastikan bahwa operator memasukkan nilai dengan benar.

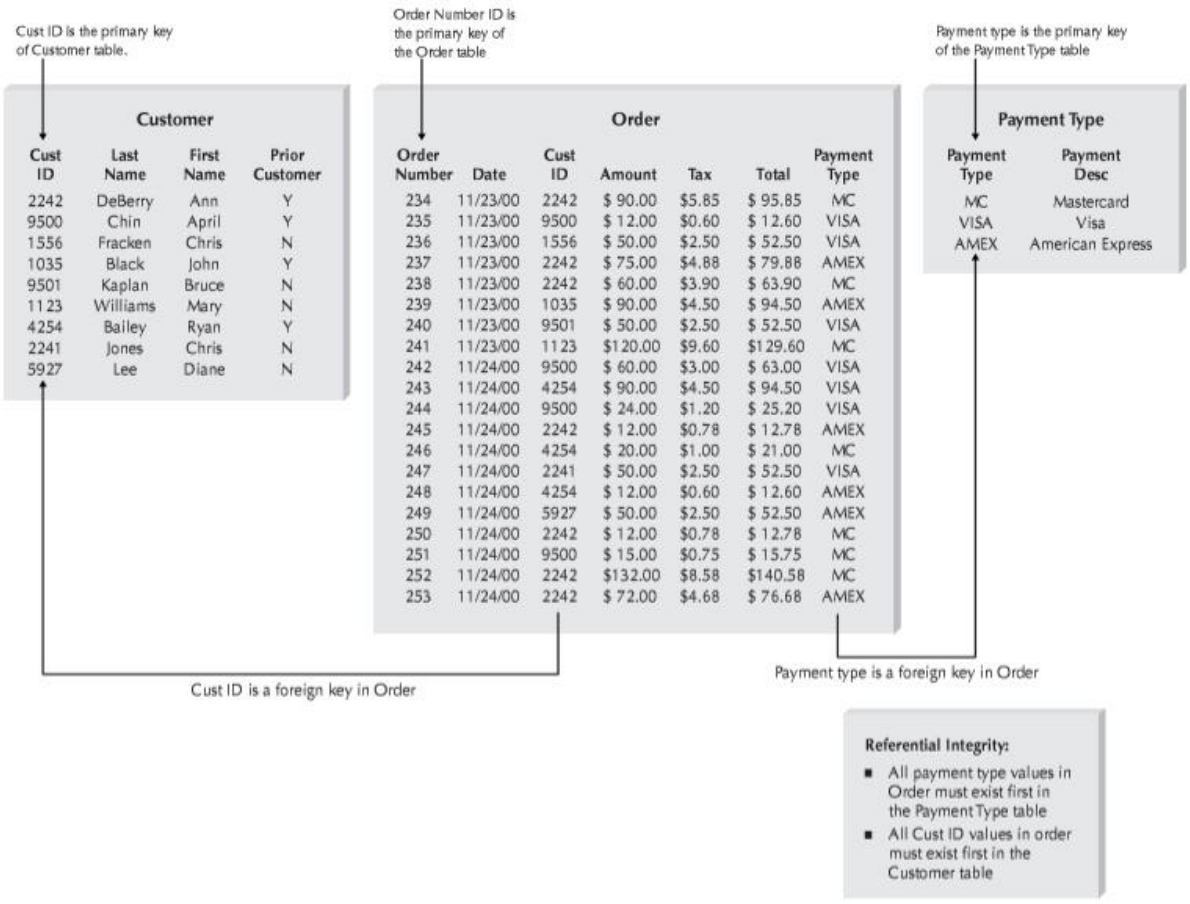
File transaksi menyimpan informasi yang dapat digunakan untuk memperbarui file master. File transaksi dapat dimusnahkan setelah perubahan ditambahkan, atau file dapat disimpan jika transaksi perlu diakses lagi di masa mendatang. Perubahan alamat pelanggan, misalnya, akan disimpan dalam file transaksi sampai program dijalankan yang memperbarui file induk alamat pelanggan dengan informasi baru.

Untuk tujuan pengendalian, perusahaan mungkin perlu menyimpan informasi tentang bagaimana data berubah dari waktu ke waktu. Misalnya, saat pegawai sumber daya manusia mengubah gaji karyawan dalam sistem sumber daya manusia, sistem harus mencatat orang yang membuat perubahan pada jumlah gaji, tanggal, dan perubahan aktual yang dibuat. File audit mencatat sebelum dan sesudah gambar data diubah sehingga audit dapat dilakukan jika integritas data dipertanyakan.

Terkadang file menjadi sangat besar sehingga menjadi berat, dan banyak informasi dalam file tidak lagi digunakan. File histori (atau file arsip) menyimpan transaksi sebelumnya (mis., pelanggan lama, pesanan sebelumnya) yang tidak lagi dibutuhkan oleh pengguna sistem. Biasanya file disimpan secara offline, namun dapat diakses sesuai kebutuhan. File lain, seperti file master, kemudian dapat disederhanakan untuk menyertakan hanya informasi yang aktif atau sangat baru.

Database Relasional

Database relasional adalah jenis database yang paling populer untuk pengembangan aplikasi saat ini. Database relasional didasarkan pada kumpulan tabel dengan setiap tabel memiliki utama utama—bidang atau bidang yang nilainya unik untuk setiap baris tabel. Tabel terkait satu sama lain dengan menempatkan utama utama dari satu tabel ke tabel terkait sebagai utama asing (lihat Gambar 9-3). Sebagian besar sistem manajemen database relasional (RDBMS) mendukung integritas referensial, atau gagasan untuk memastikan bahwa nilai yang menghubungkan tabel bersama-sama melalui utama utama dan utama asing adalah valid dan disinkronkan dengan benar. Misalnya, jika petugas entri pesanan menggunakan tabel pada Gambar 9-3 mencoba menambahkan pesanan 254 untuk nomor pelanggan 1111, dia akan membuat kesalahan karena tidak ada pelanggan di tabel Pelanggan dengan nomor itu. Jika RDBMS mendukung integritas referensial, ia akan memeriksa nomor pelanggan di tabel Pelanggan, menemukan bahwa nomor 1111 tidak valid, dan mengembalikan kesalahan ke petugas entri. Petugas kemudian akan kembali ke formulir pemesanan asli dan memeriksa kembali informasi pelanggan.



Gambar 9-3 Database Relasional

Database Obyek-Relasional

Sistem manajemen database relasional objek/*Object-relational database management systems* (ORDBMS) adalah sistem manajemen database relasional dengan ekstensi untuk menangani penyimpanan objek dalam struktur tabel relasional. Ini biasanya dilakukan melalui penggunaan tipe yang ditentukan pengguna. Misalnya, atribut dalam tabel dapat memiliki tipe data peta, yang akan mendukung penyimpanan peta. Ini adalah contoh tipe data yang kompleks. Dalam RDBMS murni, atribut terbatas pada tipe data sederhana atau atomik, seperti integer, float, atau karakter.

ORDBMS, karena mereka hanyalah ekstensi untuk rekan RDBMS mereka, mereka juga memiliki dukungan yang sangat baik untuk operasi manajemen data tipikal yang diharapkan oleh bisnis dari RDBMS, termasuk bahasa kueri/ *Structured query languag* (SQL), otorisasi, kontrol konkurensi yang mudah digunakan. , dan fasilitas pemulihan. Namun, karena SQL dirancang hanya untuk menangani tipe data sederhana, SQL juga telah diperluas untuk menangani data objek yang kompleks. Saat ini, vendor menangani masalah ini dengan cara yang berbeda. Misalnya, DB2, Informix, dan Oracle semuanya memiliki ekstensi yang menyediakan beberapa tingkat dukungan untuk objek.

Banyak ORDBMS di pasaran masih belum mendukung semua fitur berorientasi objek yang dapat muncul dalam desain berorientasi objek (mis., pewarisan). Seperti dijelaskan dalam Bab 8, salah satu masalah dalam mendukung pewarisan adalah bahwa dukungan pewarisan bergantung pada bahasa. Misalnya, cara Smalltalk mendukung pewarisan berbeda dengan pendekatan C^{++}, yang berbeda dengan pendekatan Java. Dengan demikian, vendor saat ini harus mendukung banyak versi warisan yang berbeda, satu untuk setiap bahasa berorientasi objek, atau memutuskan versi tertentu dan memaksa pengembang untuk memetakan desain

(dan implementasi) berorientasi objek mereka ke pendekatan mereka. Seperti RDBMS, pemetaan dari diagram kelas UML ke skema database relasional objek diperlukan.

Database Berorientasi Objek

Jenis sistem manajemen database berikutnya yang kami uraikan adalah sistem manajemen database berorientasi objek/*object-oriented database management systems* (OODBMS). Ada dua pendekatan utama untuk mendukung kegigihan objek dalam komunitas OODBMS: menambahkan ekstensi kegigihan ke bahasa pemrograman berorientasi objek dan membuat sistem manajemen database yang sepenuhnya terpisah.

Dengan OODBMS, koleksi objek diasosiasikan dengan suatu perluasan. Luas hanyalah kumpulan instance yang terkait dengan kelas tertentu (yaitu, setara dengan tabel di RDBMS). Secara teknis, setiap instance kelas memiliki pengidentifikasi unik yang ditetapkan oleh OODBMS: ID Objek. Namun, dari sudut pandang praktis, masih merupakan ide yang baik untuk memiliki utama utama yang bermakna secara semantik (meskipun dari perspektif OODBMS ini tidak diperlukan). Integritas referensial masih sangat penting. Dalam OODBMS, dari sudut pandang pengguna, terlihat seolah-olah objek tersebut berada di dalam objek lain. Namun, OODBMS sebenarnya melacak hubungan ini melalui penggunaan ID Objek, dan karena itu utama asing secara teknis tidak diperlukan.

OODBMS menyediakan dukungan untuk beberapa bentuk pewarisan. Namun, seperti yang telah dibahas, pewarisan cenderung bergantung pada bahasa. Saat ini, sebagian besar OODBMS terikat erat dengan bahasa pemrograman berorientasi objek/*object-oriented programming language* (OOPL) tertentu atau satu set OOPL. Awalnya, sebagian besar OODBMS mendukung Smalltalk atau C++. Saat ini, banyak OODBMS yang tersedia secara komersial menyediakan dukungan untuk C++, Java, dan Smalltalk.

OODBMS juga mendukung gagasan kelompok berulang (bidang) atau atribut multinilai. Ini didukung melalui penggunaan set atribut dan set hubungan. RDBMS tidak secara eksplisit mengizinkan atribut multinilai atau grup berulang. Ini dianggap sebagai pelanggaran terhadap bentuk normal pertama (dibahas nanti dalam bab ini) untuk database relasional. Beberapa ORDBMS mendukung grup berulang dan atribut multinilai.

Sampai saat ini, OODBMS terutama digunakan untuk mendukung aplikasi atau sistem multimedia yang melibatkan data yang kompleks (misalnya, grafik, video, suara). Area aplikasi, seperti desain dan manufaktur berbantuan komputer (CAD/CAM), layanan keuangan, sistem informasi geografis, perawatan kesehatan, telekomunikasi, dan transportasi, telah menjadi yang paling mudah menerima OODBMS. Mereka juga menjadi teknologi populer untuk mendukung perdagangan elektronik, katalog online, dan aplikasi multimedia Web yang besar. Contoh OODBMS murni termasuk Gemstone, Objectivity, db4o, dan Versant.

Meskipun OODBMS murni ada, sebagian besar organisasi saat ini berinvestasi dalam teknologi ORDBMS. Pasar untuk OODBMS diperkirakan akan tumbuh, tetapi mitra ORDBMS dan RDBMS mengerdilkannya. Salah satu alasan untuk situasi ini adalah bahwa ada banyak pengembang dan alat yang lebih berpengalaman di arena RDBMS. Selain itu, pengguna relasional menemukan bahwa menggunakan OODBMS datang dengan kurva belajar yang cukup curam.

Data Store NoSQL

Penyimpanan data NoSQL adalah jenis persistensi objek terbaru yang tersedia. Tergantung pada siapa Anda berbicara, NoSQL adalah singkatan dari No SQL atau Not Only

SQL. Terlepas dari itu, penyimpanan data yang dijelaskan sebagai NoSQL biasanya tidak mendukung SQL. Saat ini, tidak ada standar untuk penyimpanan data NoSQL. Sebagian besar penyimpanan data NoSQL dibuat untuk mengatasi masalah yang terkait dengan penyimpanan data terdistribusi dalam jumlah besar di RDBMS. Penyimpanan data NoSQL cenderung mendukung kueri yang sangat cepat. Namun, dalam hal pembaruan, penyimpanan data NoSQL biasanya tidak mendukung mekanisme penguncian, dan akibatnya, semua salinan data tidak harus konsisten setiap saat. Sebaliknya mereka cenderung mendukung model berbasis yang pada akhirnya konsisten. Jadi secara teknis dimungkinkan untuk memiliki nilai berbeda untuk salinan berbeda dari objek yang sama yang disimpan di lokasi berbeda dalam sistem terdistribusi. Tergantung pada aplikasinya, ini dapat menyebabkan masalah bagi pengambil keputusan. Oleh karena itu, penerapannya terbatas dan tidak berlaku untuk sebagian besar sistem pemrosesan transaksi bisnis tradisional. Beberapa penyimpanan data NoSQL yang lebih dikenal termasuk Tabel Besar Google, Dynamo Amazon, HBase Apache, CouchDB Apache, dan Cassandra Apache/Facebook. Ada banyak jenis penyimpanan data NoSQL, termasuk penyimpanan nilai utama, penyimpanan dokumen, penyimpanan berorientasi kolom, dan database objek. Selain database objek, yang berupa ORDBMS atau OODBMS (lihat bagian sebelumnya), kami menjelaskan setiap jenis penyimpanan data NoSQL di bawah ini.

Penyimpanan data nilai utama pada dasarnya menyediakan indeks terdistribusi (utama utama) ke tempat BLOB (objek besar biner) disimpan. BLOB memperlakukan satu set atribut sebagai satu objek besar. Contoh bagus dari jenis penyimpanan data NoSQL ini adalah Dynamo Amazon. Dynamo menyediakan dukungan untuk banyak layanan inti untuk Amazon. Jelas, sebagai salah satu situs e-commerce terbesar di dunia, Amazon membutuhkan solusi untuk persistensi objek yang dapat diskalakan, didistribusikan, dan andal. Solusi berbasis RDBMS tipikal tidak akan berfungsi untuk beberapa aplikasi ini. Aplikasi yang biasanya menggunakan penyimpanan data nilai utama adalah keranjang belanja berbasis web, katalog produk, dan daftar buku terlaris. Jenis aplikasi ini tidak memerlukan pembaruan data yang mendasarinya. Misalnya, Anda tidak memperbarui judul buku di keranjang belanja saat Anda melakukan pembelian di Amazon. Mengingat skala dan sifat terdistribusi dari jenis sistem ini, pasti ada banyak kegagalan di seluruh sistem. Menjadi toleran terhadap kesalahan dan sementara mengorbankan beberapa konsistensi di semua salinan objek adalah pertukaran yang masuk akal.

Penyimpanan data dokumen, seperti namanya, dibangun di sekitar gagasan dokumen. Ide database dokumen telah ada sejak lama. Salah satu sistem awal yang menggunakan pendekatan ini adalah Lotus Notes. Jenis toko ini dianggap bebas skema. Maksud kami tidak ada desain database yang detail. Contoh aplikasi yang baik yang akan mendapat manfaat dari jenis pendekatan ini adalah database kartu nama. Dalam database relasional, beberapa tabel perlu dirancang. Dalam penyimpanan data dokumen, desain dilakukan lebih banyak dengan cara yang “tepat waktu”. Saat kartu nama baru dimasukkan ke dalam sistem, atribut yang sebelumnya tidak disertakan hanya ditambahkan ke desain yang berkembang. Kartu nama yang dimasukkan sebelumnya tidak akan memiliki atribut yang terkait dengannya. Satu perbedaan utama antara penyimpanan data nilai utama dan penyimpanan data dokumen adalah bahwa "dokumen" memiliki struktur dan dapat dengan mudah dicari berdasarkan atribut non-utama yang terkandung dalam dokumen, sedangkan penyimpanan data nilai utama hanya memperlakukan "nilai ” sebagai satu objek monolitik besar. CouchDB Apache adalah contoh yang baik dari jenis penyimpanan data ini.

Penyimpanan data kolom mengatur data ke dalam kolom, bukan baris. Namun, tampaknya ada beberapa kebingungan tentang apa yang sebenarnya tersirat dari ini. Dalam pendekatan pertama untuk penyimpanan data kolom, baris mewakili atribut dan kolom mewakili objek. Sebaliknya, database relasional mewakili atribut dalam kolom dan mewakili

objek dalam baris. Jenis penyimpanan data kolumnar ini sangat efektif dalam intelijen bisnis, penambangan data, dan aplikasi pergudangan data di mana datanya cukup statis dan banyak perhitungan dilakukan pada satu atau sebagian kecil atribut yang tersedia. Dibandingkan dengan database relasional di mana Anda harus memilih satu set atribut dari semua baris, dengan jenis penyimpanan data ini, Anda hanya perlu memilih satu set baris. Ini harus jauh lebih cepat daripada dengan database relasional. Beberapa contoh bagus dari jenis penyimpanan data kolumnar ini termasuk *Oracle's Retail Predictive Application Server*/Server Aplikasi Prediktif Ritel Oracle, Vertica HP, dan Sybase IQ SAP. Pendekatan kedua untuk penyimpanan data kolumnar, yang mencakup HBase Apache, Cassandra Apache/Facebook, dan BigTable Google, dirancang untuk menangani kumpulan data yang sangat besar (petabyte data) yang dapat diakses seolah-olah data disimpan dalam kolom. Namun, dalam kasus ini, data sebenarnya disimpan dalam peta tiga dimensi yang terdiri dari ID objek, nama atribut, stempel waktu, dan nilai alih-alih menggunakan kolom dan baris. Pendekatan ini sangat terukur dan dapat didistribusikan. Jenis penyimpanan data ini mendukung aplikasi sosial seperti Twitter dan Facebook dan mendukung aplikasi pencarian seperti Google Maps, Earth, dan Analytics.

Mengingat popularitas komputasi sosial, intelijen bisnis, penambangan data, pergudangan data, e-commerce, dan kebutuhan mereka akan penyimpanan data yang sangat skalabel, dapat didistribusikan, dan andal, penyimpanan data NoSQL adalah area yang harus dipertimbangkan sebagai bagian dari persistensi objek. larutan. Namun, mengingat keragaman dan kompleksitas penyimpanan data NoSQL secara keseluruhan dan penerapannya yang terbatas pada aplikasi bisnis tradisional, kami tidak mempertimbangkannya lebih jauh dalam teks ini.

Memilih Format Persistensi Objek

Masing-masing format penyimpanan file dan database yang disajikan memiliki kelebihan dan kekurangannya masing-masing, dan tidak ada satu format pun yang secara inheren lebih baik dari yang lain. Bahkan, terkadang tim proyek memilih beberapa format (misalnya, database relasional untuk satu format, file untuk format lain, dan database berorientasi objek untuk format ketiga). Oleh karena itu, penting untuk memahami kekuatan dan kelemahan masing-masing format dan kapan harus menggunakannya. Gambar 9-4 menyajikan ringkasan karakteristik masing-masing dan karakteristik yang dapat membantu mengidentifikasi kapan setiap jenis format lebih sesuai.

Kekuatan dan Kelemahan Utama

Kekuatan utama dari file adalah sebagai berikut: Beberapa dukungan untuk akses sekuensial dan acak file biasanya merupakan bagian dari OOPL, file dapat dirancang untuk menjadi sangat efisien, dan mereka adalah alternatif yang baik untuk penyimpanan sementara atau jangka pendek. Namun, semua manipulasi file harus dilakukan melalui OOPL. File tidak memiliki bentuk kontrol akses apa pun di luar sistem operasi yang mendasarinya. Akhirnya, dalam banyak kasus, jika file digunakan untuk penyimpanan permanen, data yang berlebihan kemungkinan besar akan terjadi. Ini dapat menyebabkan banyak anomali pembaruan.

	File Akses Berurutan dan Acak	**DBMS relasional**	**DBMS relasional objek**	**DBMS Berorientasi Objek**	**NoSQL data store**
Kekuatan Utama	Biasanya bagian dari	Pemimpin di pasar database	Berdasarkan teknologi	Mampu menangani	Mampu menangani

	bahasa pemrogram an berorientasi objek File dapat dirancang untuk kinerja yang cepat Baik untuk penyimpana n data jangka pendek	Dapat menangani beragam kebutuhan data	terbukti yang mapan, misalnya, SQL Mampu menangani data yang kompleks	data yang kompleks Dukungan langsung untuk orientasi objek	data yang kompleks
Kelemahan Utama	Data redundan Data harus diperbarui menggunak an program, mis., tidak ada manipulasi atau bahasa kueri Tidak ada kontrol akses	Tidak dapat menangani data yang kompleks Tidak ada dukungan untuk orientasi objek Ketidakcocok an impedansi antara tabel dan objek	Dukungan terbatas untuk orientasi objek Ketidakcocok an impedansi antara tabel dan objek	Teknologi masih matang Keterampil an sulit ditemukan	Teknologi masih matang Keterampil an sulit ditemukan
Tipe Data yang Didukung	Sederhana dan Kompleks	Sederhana	Sederhana dan Kompleks	Sederhana dan Kompleks	Sederhana dan Kompleks
Jenis Sistem Aplikasi yang Didukung	Pemrosesan transaksi	Pemrosesan transaksi dan pengambilan keputusan	Pemrosesan transaksi dan pengambilan keputusan	Pemrosesan transaksi dan pengambila n keputusan	Terutama pengambila n keputusan
Format Penyimpan an yang Ada	Tergantung organisasi	Tergantung organisasi	Tergantung organisasi	Tergantung organisasi	Tergantung organisasi
Kebutuhan Masa Depan	Prospek masa depan yang buruk	Prospek masa depan yang baik	Prospek masa depan yang baik	Prospek masa depan yang baik	Prospek masa depan yang baik

Gambar 9-4 Perbandingan Format Persistensi Objek

RDBMS membawa serta teknologi komersial yang telah terbukti. Mereka adalah pemimpin di pasar DBMS. Selain itu, mereka dapat menangani kebutuhan data yang sangat beragam. Namun, mereka tidak dapat menangani tipe data yang kompleks, seperti gambar. Oleh karena itu, semua objek harus dikonversi ke bentuk yang dapat disimpan dalam tabel yang terdiri dari data atomik atau sederhana. Mereka tidak memberikan dukungan untuk orientasi objek. Kurangnya dukungan ini menyebabkan ketidakcocokan impedansi antara

objek yang terkandung dalam OOPL dan data yang disimpan dalam tabel. Ketidakcocokan impedansi mengacu pada jumlah pekerjaan yang dilakukan oleh pengembang dan DBMS dan potensi kehilangan informasi yang dapat terjadi saat mengonversi objek ke formulir yang dapat disimpan dalam tabel.

Karena ORDBMS biasanya merupakan ekstensi berorientasi objek ke RDBMS, mereka mewarisi kekuatan RDBMS. Mereka didasarkan pada teknologi mapan, seperti SQL, dan tidak seperti pendahulunya, mereka dapat menangani tipe data yang kompleks. Namun, mereka hanya menyediakan dukungan terbatas untuk orientasi objek. Tingkat dukungan bervariasi di antara vendor; oleh karena itu, ORDBMS juga mengalami masalah ketidakcocokan impedansi.

OODBMS mendukung tipe data yang kompleks dan memiliki keuntungan mendukung orientasi objek secara langsung. Oleh karena itu, mereka tidak mengalami ketidakcocokan impedansi seperti yang dilakukan DBMS sebelumnya. Namun, komunitas OODBMS masih dalam tahap pendewasaan. Oleh karena itu, teknologi ini mungkin masih terlalu berisiko bagi beberapa perusahaan. Masalah utama lainnya dengan OODBMS adalah kurangnya tenaga kerja terampil dan kurva pembelajaran yang dirasakan oleh komunitas RDBMS. Penyimpanan data NoSQL mendukung tipe data yang kompleks. Namun, mereka dapat mengalami beberapa bentuk ketidakcocokan impedansi. Masalah utama dengan penyimpanan data NoSQL adalah kurangnya kedewasaan dan kurangnya tenaga kerja terampil yang tahu cara menggunakannya secara efektif.

Jenis Data yang Didukung Masalah pertama adalah jenis data yang perlu disimpan dalam sistem. Sebagian besar aplikasi perlu menyimpan tipe data sederhana, seperti teks, tanggal, dan angka. Semua file dan DBMS dilengkapi untuk menangani jenis data ini. Namun, pilihan terbaik untuk penyimpanan data sederhana biasanya adalah RDBMS karena teknologinya telah matang dari waktu ke waktu dan terus ditingkatkan untuk menangani data sederhana dengan sangat efektif.

Semakin banyak, aplikasi menggabungkan data yang kompleks, seperti video, gambar, atau audio. ORDBMS, OODBMS, atau penyimpanan data NoSQL paling mampu menangani data jenis ini. Data kompleks yang disimpan sebagai objek dapat dimanipulasi lebih cepat dibandingkan dengan format penyimpanan lainnya.

Jenis Sistem Aplikasi Ada banyak jenis sistem aplikasi yang dapat dikembangkan. Sistem pemrosesan transaksi dirancang untuk menerima dan memproses banyak permintaan simultan (misalnya, entri pesanan, distribusi, penggajian). Dalam sistem pemrosesan transaksi, data terus diperbarui oleh sejumlah besar pengguna, dan kueri yang dibutuhkan sistem ini biasanya ditentukan sebelumnya atau ditargetkan pada subset kecil catatan (mis., Daftar pesanan yang dipesan sebelumnya hari ini atau Produk apa apakah pelanggan #1234 memesan pada 12 Mei 2001?).

Kumpulan sistem aplikasi lainnya adalah kumpulan yang dirancang untuk mendukung pengambilan keputusan, seperti sistem pendukung keputusan/ *decision support systems* (DSS), sistem informasi manajemen//*management information systems* (MIS), sistem informasi eksekutif */executive information systems* (EIS), dan sistem pakar/ *expert systems* (ES). Sistem pendukung pengambilan keputusan ini dibangun untuk mendukung pengguna yang perlu memeriksa sejumlah besar data historis hanya-baca. Pertanyaan yang mereka ajukan sering kali bersifat ad hoc, dan mencakup ratusan atau ribuan catatan sekaligus (misalnya, Daftar semua pelanggan di wilayah Barat yang membeli produk seharga lebih dari Rp. 7.500.000 setidaknya tiga kali, atau Produk apa yang telah meningkatkan penjualan di bulan-bulan musim panas yang belum diklasifikasikan sebagai barang dagangan musim panas?).

Oleh karena itu, sistem pemrosesan transaksi dan DSS memiliki kebutuhan penyimpanan data yang sangat berbeda. Sistem pemrosesan transaksi memerlukan format penyimpanan data yang disesuaikan untuk banyak pembaruan data dan pengambilan cepat pertanyaan spesifik yang telah ditentukan sebelumnya. File, database relasional, database relasional objek, dan database berorientasi objek semuanya dapat mendukung persyaratan semacam ini. Sebaliknya, sistem untuk mendukung pengambilan keputusan biasanya hanya membaca data (tidak memperbaruinya), seringkali secara ad hoc. Pilihan terbaik untuk sistem ini biasanya adalah RDBMS karena format ini dapat dikonfigurasi khusus untuk kebutuhan yang mungkin tidak jelas dan kurang tepat untuk mengubah data. Namun, tergantung pada jenis data yang dibutuhkan untuk mendukung aplikasi pengambilan keputusan, RDBMS mungkin tidak sesuai. Dalam hal ini, ORDBMS, OODBMS, atau penyimpanan data NoSQL mungkin merupakan solusi yang lebih baik.

Format Penyimpanan yang Ada. Format penyimpanan harus dipilih terutama berdasarkan jenis data dan sistem aplikasi yang dikembangkan. Namun, tim proyek harus mempertimbangkan format penyimpanan yang ada di organisasi saat membuat keputusan desain. Dengan cara ini, mereka dapat lebih memahami keterampilan teknis yang sudah ada dan seberapa curam kurva pembelajarannya ketika format penyimpanan diadopsi. Misalnya, perusahaan yang akrab dengan RDBMS akan memiliki sedikit masalah dalam mengadopsi database relasional untuk proyek tersebut, sedangkan OODBMS atau penyimpanan data NoSQL mungkin memerlukan pelatihan pengembang yang substansial.

Kebutuhan Masa Depan. Tim proyek tidak hanya harus mempertimbangkan teknologi penyimpanan di dalam perusahaan, tetapi juga harus menyadari tren dan teknologi saat ini yang digunakan oleh organisasi lain. Sejumlah besar penginstalan jenis format penyimpanan tertentu menunjukkan bahwa keterampilan dan produk tersedia untuk mendukung format tersebut. Oleh karena itu, pemilihan format tersebut aman. Misalnya, mungkin akan lebih mudah dan lebih murah untuk menemukan keahlian RDBMS saat menerapkan sistem daripada mencari bantuan dengan OODBMS atau penyimpanan data NoSQL.

Kriteria Lain-lain. Kriteria lain yang harus dipertimbangkan termasuk biaya, masalah lisensi, kontrol konkurensi, kemudahan penggunaan, kontrol keamanan dan akses, manajemen versi, manajemen penyimpanan, manajemen utama, manajemen kueri, pengikatan bahasa, dan API. Kami juga harus mempertimbangkan masalah kinerja, seperti manajemen cache, penyisipan, penghapusan, pengambilan, dan pembaruan objek kompleks. Akhirnya, tingkat dukungan untuk orientasi objek (seperti objek, pewarisan tunggal, pewarisan berganda, polimorfisme, enkapsulasi dan penyembunyian informasi, metode, atribut multinilai, grup berulang) sangat penting.

9.4 MASALAH PEMETAAN OBJEK DOMAIN KE FORMAT PERSISTENSI OBJEK

Ada banyak format berbeda yang dapat dipilih untuk mendukung persistensi objek. Setiap format dapat memiliki beberapa persyaratan konversi. Terlepas dari format persistensi objek yang dipilih, kami menyarankan untuk mendukung utama utama dan utama asing dengan menambahkannya ke kelas domain masalah pada saat ini. Namun, ini menyiratkan bahwa beberapa pemrosesan tambahan akan diperlukan. Pengembang harus menetapkan nilai untuk utama asing saat menambahkan hubungan ke objek. Dari perspektif praktis, format file sebagian besar digunakan untuk penyimpanan sementara. Jadi, kami tidak mempertimbangkannya lebih lanjut.

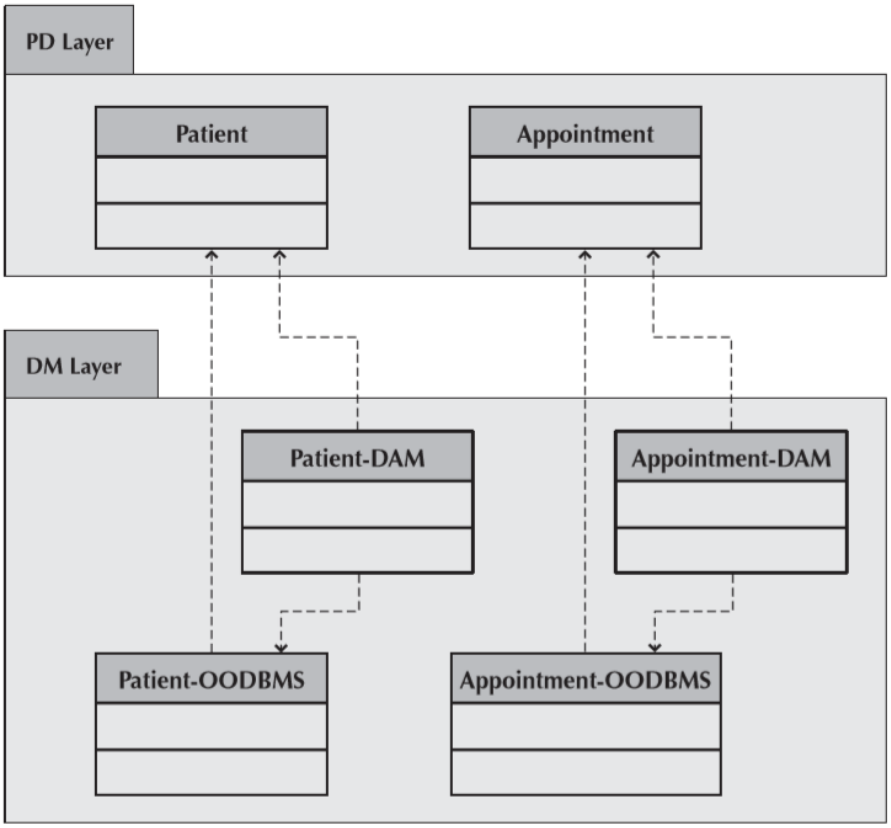
Kami juga menyarankan agar fungsionalitas manajemen data spesifik, seperti pengambilan dan pembaruan data dari penyimpanan objek, disertakan hanya dalam kelas yang terdapat dalam lapisan manajemen data. Ini akan memastikan bahwa kelas manajemen data

bergantung pada kelas domain masalah dan bukan sebaliknya. Hal ini memungkinkan desain kelas domain masalah menjadi independen dari lingkungan persistensi objek tertentu, sehingga meningkatkan portabilitas dan potensinya untuk digunakan kembali. Seperti rekomendasi kami sebelumnya, ini juga menyiratkan pemrosesan tambahan.

Memetakan Objek Domain Masalah ke Format OODBMS

Jika kita mendukung persistensi objek dengan OODBMS, pemetaan antara objek domain masalah dan OODBMS cenderung cukup mudah. Sebagai titik awal, kami menyarankan bahwa setiap kelas domain masalah konkret harus memiliki kelas persistensi objek yang sesuai di OODBMS. Juga akan ada kelas akses dan manipulasi data/*data access and manipulation* (DAM) (dijelaskan nanti dalam bab ini) yang berisi fungsionalitas yang diperlukan untuk mengelola interaksi antara kelas persistensi objek dan lapisan domain masalah. Misalnya, menggunakan contoh sistem janji temu dari bab-bab sebelumnya, kelas Pasien dikaitkan dengan kelas OODBMS (lihat Gambar 9-5). Kelas Pasien pada dasarnya tidak akan berubah dari analisis. Kelas Pasien-OODBMS akan menjadi kelas baru yang bergantung pada kelas Pasien, sedangkan kelas Pasien-DAM akan menjadi kelas baru yang bergantung pada kelas Pasien dan kelas PasienOODBMS. Kelas Patient-DAM harus dapat membaca dan menulis ke OODBMS. Jika tidak, itu tidak akan dapat menyimpan dan mengambil instance dari kelas Pasien. Meskipun ini menambah overhead pada instalasi sistem, ini memungkinkan kelas domain masalah menjadi independen dari OODBMS yang digunakan. Jika di lain waktu OODBMS atau format persistensi objek diadopsi, hanya kelas DAM yang harus dimodifikasi. Pendekatan ini meningkatkan portabilitas dan potensi penggunaan kembali kelas domain masalah.

Meskipun kami mengimplementasikan lapisan DAM menggunakan OODBMS, pemetaan dari lapisan domain masalah ke kelas OODBMS di lapisan akses data dan manajemen mungkin diperlukan. Jika pewarisan berganda digunakan dalam domain masalah tetapi tidak didukung oleh OODBMS, maka pewarisan berganda harus difaktorkan dari kelas-kelas OODBMS. Untuk setiap kasus pewarisan berganda (yaitu, lebih dari satu superclass), aturan berikut dapat digunakan untuk memfaktorkan efek pewarisan berganda dalam desain kelas OODBMS.



Gambar 9-5 Masalah Sistem Pengangkatan Domain dan Lapisan DM

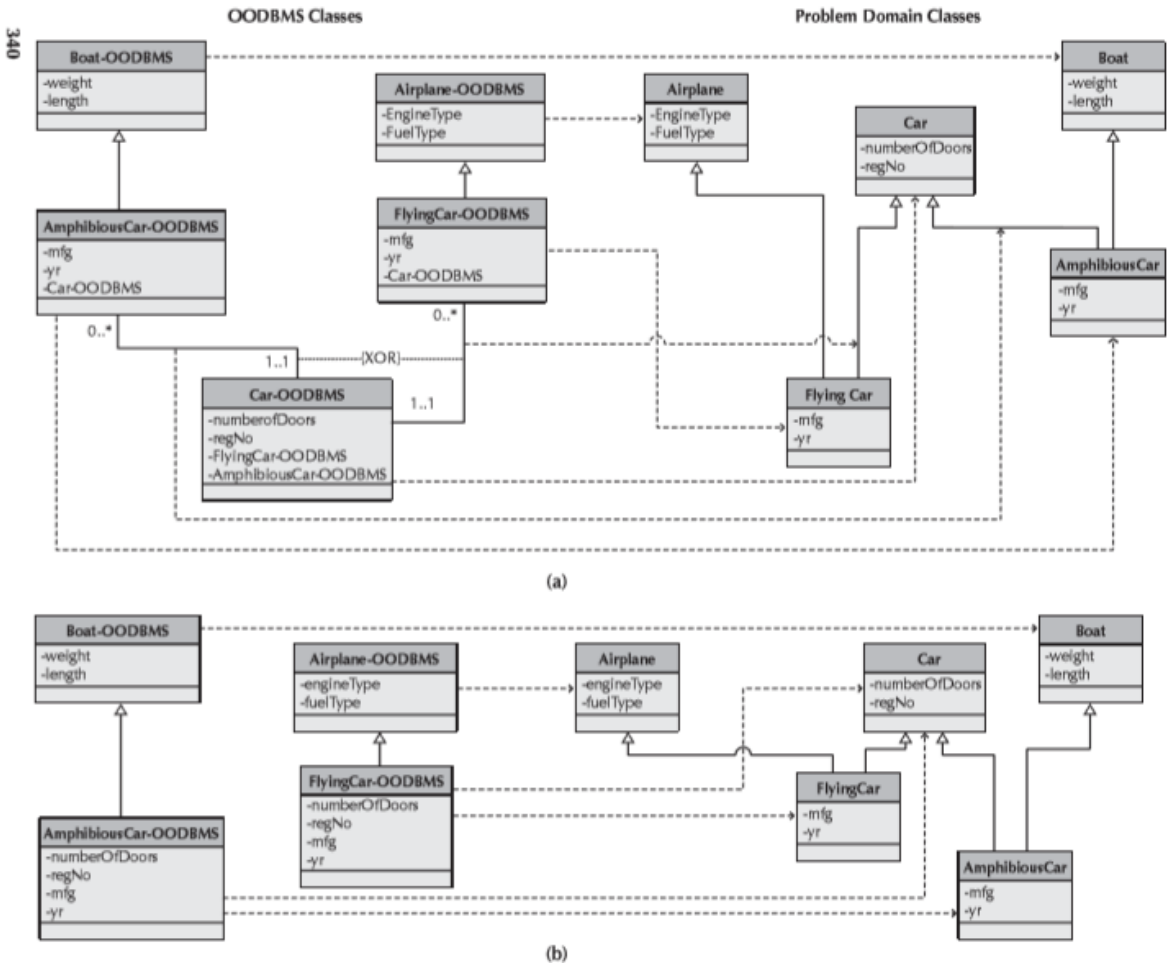
Aturan 1a:

Tambahkan kolom ke kelas OODBMS yang mewakili subkelas yang akan berisi ID Objek dari instance yang disimpan di kelas OODBMS yang mewakili superkelas "tambahan". Ini mirip dalam konsep dengan utama asing dalam RDBMS. Multiplisitas dari asosiasi baru ini dari subclass ke "superclass" harus 1..1. Tambahkan kolom ke kelas OODBMS yang mewakili superkelas yang akan berisi ID Objek dari instance yang disimpan di kelas OODBMS yang mewakili subkelas. Jika superclass adalah beton, yaitu mereka dapat dipakai sendiri, maka multiplisitas dari superclass ke subclass adalah 0..1, jika tidak, itu adalah 1..1. Batasan eksklusif-atau (XOR) harus ditambahkan di antara asosiasi. Lakukan ini untuk setiap superclass "tambahan".

atau

Aturan 1b:

Ratakan hierarki pewarisan kelas OODBMS dengan menyalin atribut dan metode dari superkelas OODBMS tambahan ke semua subkelas OODBMS dan menghapus superkelas tambahan dari desain.



Gambar 9-6 Memetakan Objek Domain Masalah ke OODBMS Berbasis Warisan Tunggal

Aturan pewarisan berganda ini sangat mirip dengan yang dijelaskan dalam Bab 8. Gambar 9-6 menunjukkan penerapan aturan ini. Sisi kanan gambar menggambarkan kelas domain masalah yang sama yang ada di Bab 8: Pesawat Terbang, Mobil, Perahu, Mobil Terbang, dan Mobil Amfibi. FlyingCar mewarisi dari Airplane dan Car, dan AmphibiousCar mewarisi dari Car dan Boat. Gambar 9-6a menggambarkan pemetaan hubungan pewarisan berganda ke dalam OODBMS berbasis pewarisan tunggal menggunakan Aturan 1a. Dengan asumsi bahwa Mobil adalah beton, kami menerapkan Aturan 1a ke kelas Domain Masalah, dan kami berakhir dengan kelas OODBMS di sisi kiri Bagian a, di mana kami memiliki:

- A menambahkan kolom (atribut) ke FlyingCar-OODBMS yang mewakili asosiasi dengan Car-OODBMS;
- Menambahkan kolom (atribut) pada OODBMS Mobil Amfibi yang merepresentasikan asosiasi dengan Mobil-OODBMS;
- A menambahkan sepasang kolom (atribut) ke Car-OODBMS yang mewakili asosiasi dengan FlyingCar-OODBMS dan AmphibiousCar-OODBMS dan demi kelengkapan;
- Asosiasi yang ditambahkan antara AmphibiousCar-OODBMS dan Car-OODBMS dan FlyingCar-OODBMS dan Car-OODBMS yang memiliki multiplisitas yang benar dan batasan XOR yang ditampilkan secara eksplisit.

Kami juga menampilkan hubungan ketergantungan dari kelas OODBMS ke kelas domain masalah. Selanjutnya, kami mengilustrasikan fakta bahwa hubungan antara FlyingCarOODBMS dan Car-OODBMS dan hubungan antara AmphibiousCar-OODBMS

dan Car-OODBMS didasarkan pada hubungan pewarisan faktor-out asli dalam kelas domain masalah dengan menunjukkan hubungan ketergantungan dari asosiasi ke hubungan warisan.

Di sisi lain, jika kita menerapkan Aturan 1b untuk memetakan kelas Domain Masalah ke OODBMS berbasis warisan tunggal, kita berakhir dengan pemetaan pada Gambar 9-6b, di mana semua atribut Mobil telah disalin ke FlyingCar-OODBMS dan Kelas AmphibiousCar-OODBMS. Dalam kasus terakhir ini, Anda mungkin harus berurusan dengan efek konflik pewarisan (lihat Bab 8).

Keuntungan dari Aturan 1a adalah bahwa semua kelas domain masalah yang diidentifikasi selama analisis disimpan dalam database. Hal ini memungkinkan fleksibilitas maksimum pemeliharaan desain lapisan manajemen data. Namun, Aturan 1a meningkatkan jumlah pengiriman pesan yang diperlukan dalam sistem, dan telah menambahkan persyaratan pemrosesan yang melibatkan batasan XOR, sehingga mengurangi efisiensi desain secara keseluruhan. Rekomendasi kami adalah untuk membatasi Aturan 1a untuk diterapkan hanya ketika berhadapan dengan superclass "ekstra" yang konkret karena mereka memiliki keberadaan independen dalam domain masalah. Gunakan Aturan 1b ketika mereka abstrak karena mereka tidak memiliki keberadaan independen dari subkelas.

Dalam kedua kasus, pemrosesan tambahan akan diperlukan. Dalam kasus pertama, cascading penghapusan akan bekerja, tidak hanya dari objek individual ke semua elemennya tetapi juga dari instance superclass ke semua instance subclass. Dalam kasus kedua, akan ada banyak penyalinan dan penempelan struktur superclass ke subclass. Dalam hal modifikasi struktur superclass diperlukan, modifikasi harus dilakukan secara cascade ke semua subclass. Namun, pewarisan berganda jarang terjadi di sebagian besar masalah bisnis. Dalam kebanyakan situasi, aturan sebelumnya tidak akan pernah diperlukan.

Saat membuat instance objek domain masalah dari objek OODBMS, pemrosesan tambahan juga akan diperlukan. Pemrosesan tambahan akan dilakukan dalam pengambilan objek OODBMS dan mengambil elemennya untuk membuat objek domain masalah. Juga, ketika menyimpan objek domain masalah, konversi ke satu set objek OODBMS diperlukan. Pada dasarnya, setiap kali terjadi interaksi antara OODBMS dan sistem, jika multiple inheritance terlibat dan OODBMS hanya mendukung single inheritance, konversi antara kedua format akan diperlukan.

Memetakan Objek Domain Masalah ke Format ORDBMS

Jika kita mendukung persistensi objek dengan ORDBMS, maka pemetaan dari objek domain masalah ke objek manajemen data jauh lebih terlibat. Tergantung pada tingkat dukungan untuk orientasi objek, aturan pemetaan yang berbeda diperlukan. Untuk tujuan kami, kami berasumsi bahwa ORDBMS mendukung ID Objek, atribut multinilai, dan prosedur tersimpan. Namun, kami berasumsi bahwa ORDBMS tidak memberikan dukungan apa pun untuk pewarisan. Berdasarkan asumsi ini, Gambar 9-7 mencantumkan seperangkat aturan yang dapat digunakan untuk merancang pemetaan dari objek Domain Masalah ke tabel lapisan manajemen data berbasis ORDBMS.

Pertama, semua kelas Domain Masalah konkret harus dipetakan ke tabel di ORDBMS. Misalnya pada Gambar 9-8, kelas Pasien telah dipetakan ke tabel Patient-ORDBMS. Perhatikan bahwa kelas Participant juga telah dipetakan ke tabel ORDBMS. Meskipun kelas Partisipan bersifat abstrak, pemetaan ini dilakukan karena pada diagram kelas lengkap (lihat Gambar 7-15), kelas Partisipan memiliki beberapa subkelas langsung (Karyawan dan Pasien).

Kedua, atribut bernilai tunggal harus dipetakan ke kolom dalam tabel ORDBMS. Sekali lagi, mengacu pada Gambar 9-8, kita melihat bahwa atribut jumlah dari kelas Pasien telah dimasukkan ke dalam kelas Tabel Pasien.

Ketiga, tergantung pada tingkat dukungan prosedur tersimpan, metode dan atribut turunan harus dipetakan ke prosedur tersimpan atau modul program.

Keempat, hubungan agregasi dan asosiasi bernilai tunggal (satu-ke-satu) harus dipetakan ke kolom yang dapat menyimpan ID Objek. Hal ini harus dilakukan untuk kedua sisi hubungan.

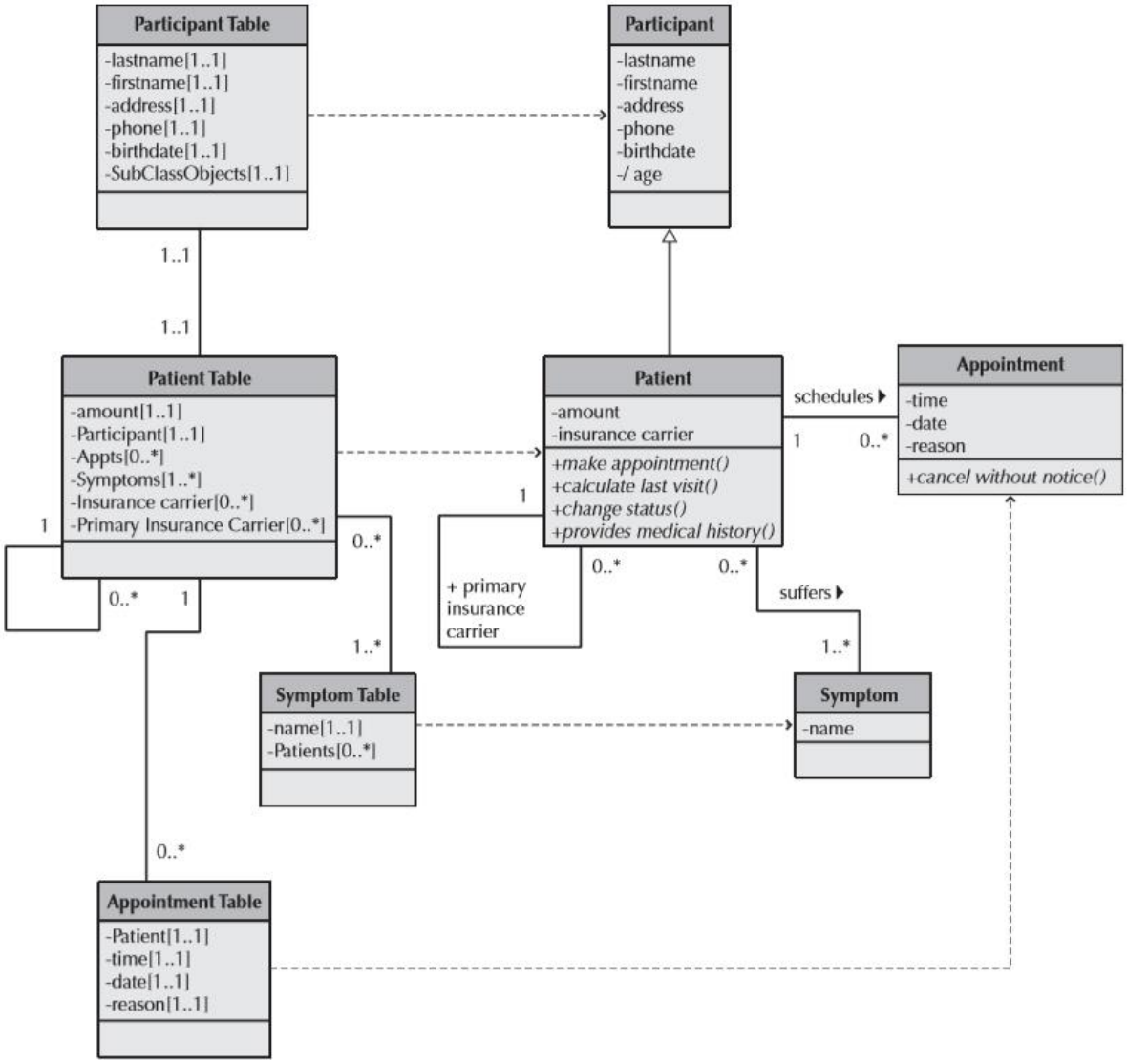
Kelima, atribut multinilai harus dipetakan ke kolom yang dapat berisi sekumpulan nilai. Misalnya pada Gambar 9-8, atribut pembawa asuransi di kelas Pasien mungkin berisi beberapa nilai karena pasien mungkin memiliki lebih dari satu operator asuransi. Jadi dalam tabel Pasien, multiplisitas telah ditambahkan ke atribut pembawa asuransi untuk menggambarkan fakta ini.

Aturan 1: Petakan semua kelas Domain Masalah konkret ke tabel ORDBMS. Juga, jika kelas domain masalah abstrak memiliki beberapa subkelas langsung, petakan kelas abstrak ke tabel ORDBMS.
Aturan 2: Petakan atribut bernilai tunggal ke kolom tabel ORDBMS.
Aturan 3: Memetakan metode dan atribut turunan ke prosedur tersimpan atau ke modul program.
Aturan 4: Memetakan agregasi dan hubungan asosiasi bernilai tunggal ke kolom yang dapat menyimpan ID Objek. Lakukan ini untuk kedua sisi hubungan.
Aturan 5: Petakan atribut multinilai ke kolom yang dapat berisi sekumpulan nilai.
Aturan 6: Petakan grup atribut yang berulang ke tabel baru dan buat asosiasi satu-ke-banyak dari tabel asli ke tabel baru.
Aturan 7: Memetakan agregasi multinilai dan hubungan asosiasi ke kolom yang dapat menyimpan sekumpulan ID Objek. Lakukan ini untuk kedua sisi hubungan.
Aturan 8: Untuk hubungan agregasi dan asosiasi dari tipe campuran (satu-ke-banyak atau banyak-ke-satu), pada sisi bernilai tunggal (1..1 atau 0..1) dari hubungan, tambahkan kolom yang dapat menyimpan satu set ID Objek. Nilai yang terkandung dalam kolom baru ini akan menjadi ID Objek dari instance kelas di sisi multinilai. Di sisi multinilai (1..* atau 0..*), tambahkan kolom yang dapat menyimpan ID Objek tunggal yang akan berisi nilai instance kelas di sisi bernilai tunggal.
Untuk hubungan generalisasi/pewarisan:
Aturan 9a: Tambahkan kolom ke tabel yang mewakili subkelas yang akan berisi ID Objek dari instance yang disimpan dalam tabel yang mewakili kelas super. Ini mirip dalam konsep dengan utama asing dalam RDBMS. Multiplisitas dari asosiasi baru ini dari subclass ke "superclass" harus 1..1. Tambahkan kolom ke tabel yang mewakili superclass yang akan berisi ID Objek dari instance yang disimpan dalam tabel yang mewakili subclass. Jika superclass adalah beton, yaitu mereka dapat dipakai sendiri, maka multiplisitas dari superclass ke subclass adalah 0..1, jika tidak, itu adalah 1..1. Batasan eksklusif-atau (XOR) harus ditambahkan di antara asosiasi. Lakukan ini untuk setiap superclass.
atau
Aturan 9b: Ratakan hierarki pewarisan dengan menyalin atribut superclass ke semua subclass dan hapus superclass dari desain.*

*Ada baiknya juga untuk mendokumentasikan modifikasi ini dalam desain sehingga di masa depan, modifikasi desain dapat dipertahankan dengan mudah.

Gambar 9-7 Skema Pemetaan Objek Domain Masalah ke ORDBMS

Aturan pemetaan keenam membahas kelompok atribut yang berulang dalam objek domain masalah. Dalam hal ini, kelompok atribut yang berulang harus digunakan untuk membuat tabel baru di ORDBMS. Ini dapat menyiratkan kelas yang hilang di lapisan domain masalah. Biasanya, saat sekumpulan atribut berulang bersama sebagai sebuah grup, itu menyiratkan kelas baru. Terakhir, kita harus membuat asosiasi satu-ke-banyak dari tabel asli ke tabel baru.



Gambar 9-8 Contoh Pemetaan Objek Domain Masalah ke Skema ORDBMS

Aturan ketujuh mendukung pemetaan agregasi multinilai (banyak ke banyak) dan hubungan asosiasi ke kolom yang dapat menyimpan sekumpulan ID Objek. Pada dasarnya, ini adalah kombinasi dari aturan keempat dan kelima. Seperti aturan keempat, ini harus dilakukan untuk kedua sisi hubungan. Misalnya pada Gambar 9-8, tabel Gejala memiliki atribut multinilai (Pasien) yang dapat berisi beberapa ID Objek ke objek Tabel Pasien, dan tabel Pasien memiliki atribut multinilai (Gejala) yang dapat berisi beberapa ID Objek ke objek Tabel Gejala.

Aturan kedelapan menggabungkan maksud Aturan 4 dan 7. Dalam hal ini, aturan memetakan hubungan satu-ke-banyak dan banyak-ke-satu. Di sisi bernilai tunggal (1..1 atau 0..1) dari hubungan, kolom yang dapat menyimpan sekumpulan ID Objek dari tabel di sisi

multinilai (1..* atau 0..*) dari hubungan harus ditambahkan. Di sisi multinilai, kolom harus ditambahkan ke tabel yang dapat menyimpan ID Objek dari instance yang disimpan dalam tabel di sisi hubungan bernilai tunggal. Misalnya, pada Gambar 9-8, tabel Pasien memiliki atribut multinilai (Appts) yang dapat berisi beberapa ID Objek ke objek Tabel Pengangkatan, sedangkan tabel Pengangkatan memiliki atribut bernilai tunggal (Pasien) yang dapat berisi ID Objek untuk objek Tabel Pasien.

Aturan kesembilan, dan terakhir, berkaitan dengan kurangnya dukungan untuk generalisasi dan pewarisan. Dalam hal ini, ada dua pendekatan yang berbeda. Pendekatan ini hampir identik dengan aturan yang dijelaskan dengan format persistensi objek OODBMS sebelumnya. Misalnya pada Gambar 9-8, tabel Pasien berisi atribut (Peserta) yang dapat berisi ID Objek untuk objek Tabel Peserta, dan tabel Peserta berisi atribut (SubClassObjects) yang berisi ID Objek untuk objek, dalam hal ini kasus, disimpan dalam tabel Pasien. Dalam kasus lain, hierarki pewarisan diratakan.

Tentu saja, pemrosesan tambahan diperlukan setiap kali terjadi interaksi antara database dan sistem. Setiap kali objek harus dibuat atau diambil dari database, diperbarui, atau dihapus, objek ORDBMS harus dikonversi ke objek domain masalah, atau sebaliknya. Satu-satunya pilihan lain adalah memodifikasi objek domain masalah. Namun, modifikasi tersebut dapat menyebabkan masalah antara lapisan domain masalah dan arsitektur fisik dan lapisan antarmuka manusia-komputer. Secara umum, biaya konversi antara ORDBMS dan lapisan domain masalah harus lebih dari diimbangi oleh penghematan waktu pengembangan yang terkait dengan interaksi antara domain masalah dan arsitektur fisik dan lapisan interaksi manusia-komputer dan kemudahan pemeliharaan lapisan domain masalah yang bersih secara semantik.

Memetakan Objek Domain Masalah ke Format RDBMS

Jika kita mendukung persistensi objek dengan RDBMS, maka pemetaan dari objek domain masalah ke tabel RDBMS mirip dengan pemetaan ke ORDBMS. Namun, asumsi yang dibuat untuk ORDBMS tidak lagi valid. Gambar 9-9 mencantumkan seperangkat aturan yang dapat digunakan untuk merancang pemetaan dari objek domain masalah ke tabel lapisan manajemen data berbasis RDBMS.

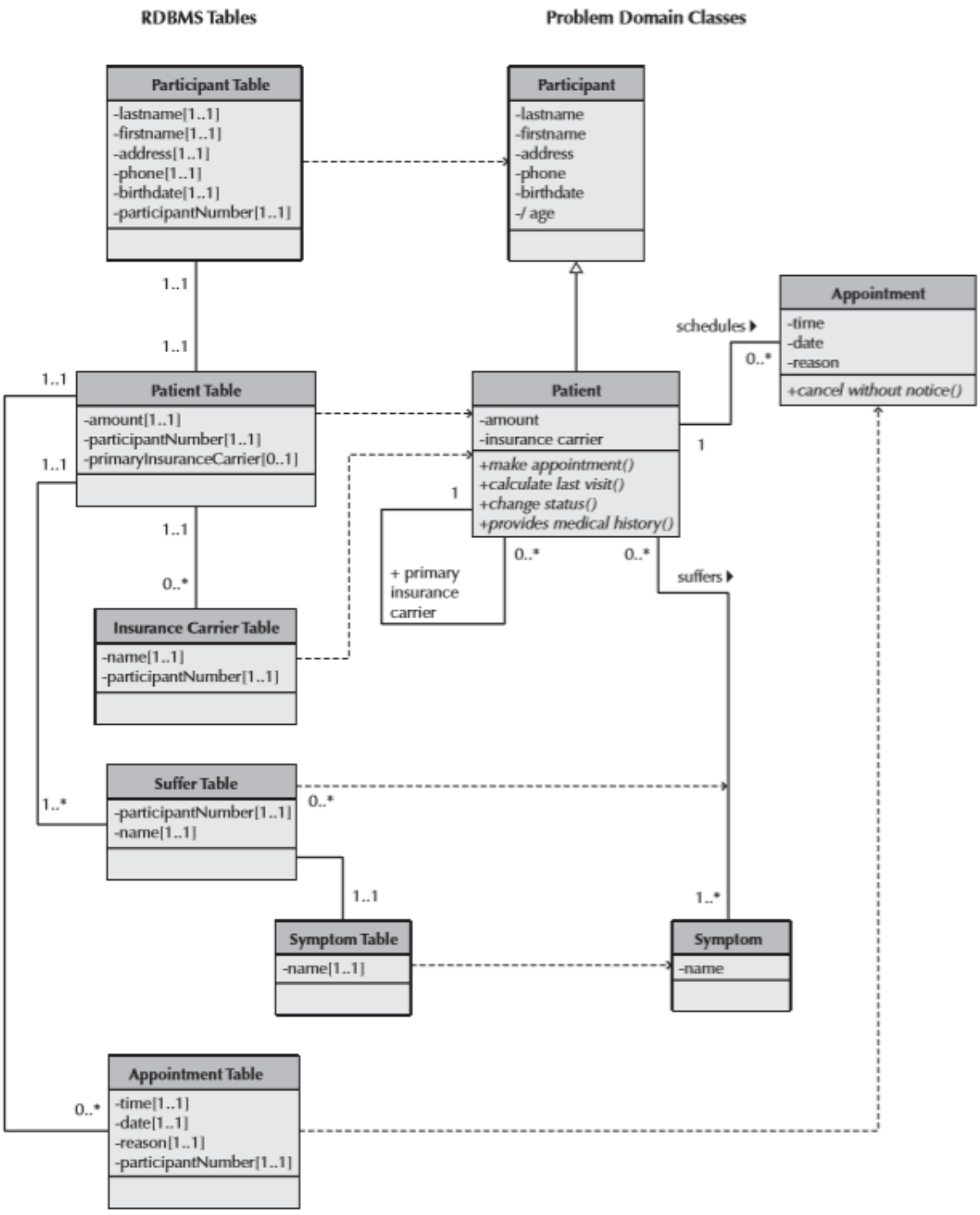
Empat aturan pertama pada dasarnya adalah seperangkat aturan yang sama yang digunakan untuk memetakan objek domain masalah ke objek manajemen data berbasis ORDBMS. Pertama, semua kelas domain masalah konkret harus dipetakan ke tabel di RDBMS. Kedua, atribut bernilai tunggal harus dipetakan ke kolom dalam tabel RDBMS. Ketiga, metode harus dipetakan ke prosedur tersimpan atau modul program, tergantung pada kompleksitas metode. Keempat, hubungan agregasi dan asosiasi bernilai tunggal (satu-ke-satu) dipetakan ke kolom yang dapat menyimpan utama asing dari tabel terkait. Ini harus dilakukan untuk kedua sisi hubungan. Misalnya pada Gambar 9-10, kita perlu menyertakan tabel dalam RDBMS untuk kelas Peserta, Pasien, Gejala, dan Janji Temu.

Aturan 1: Petakan semua kelas domain masalah konkret ke tabel RDBMS. Juga, jika kelas Domain Masalah abstrak memiliki beberapa subkelas langsung, petakan kelas abstrak ke tabel RDBMS.
Aturan 2: Petakan atribut bernilai tunggal ke kolom tabel.
Aturan 3: Petakan metode ke prosedur tersimpan atau ke modul program.
Aturan 4: Memetakan hubungan agregasi dan asosiasi bernilai tunggal ke kolom yang dapat menyimpan utama dari tabel terkait, yaitu menambahkan utama asing ke tabel. Lakukan ini untuk kedua sisi hubungan.
Aturan 5: Petakan atribut multinilai dan grup berulang ke tabel baru dan buat asosiasi satu-ke-banyak dari tabel asli ke tabel baru.

Aturan 6: Memetakan agregasi multinilai dan hubungan asosiasi ke tabel asosiatif baru yang menghubungkan dua tabel asli bersama-sama. Salin utama utama dari kedua tabel asli ke tabel asosiatif baru, yaitu, tambahkan utama asing ke tabel.
Aturan 7: Untuk hubungan agregasi dan asosiasi tipe campuran, salin utama utama dari sisi bernilai tunggal (1..1 atau 0..1) dari hubungan ke kolom baru dalam tabel di sisi multinilai (1. .* atau 0..*) dari relasi yang dapat menyimpan utama dari tabel terkait, yaitu, menambahkan utama asing ke tabel pada sisi multinilai dari relasi tersebut.
Untuk hubungan generalisasi/pewarisan:
Aturan 8a: Pastikan bahwa primary key dari instance subclass sama dengan primary key dari superclass. Multiplisitas dari asosiasi baru ini dari subclass ke "superclass" harus 1..1. Jika superclass adalah beton, yaitu mereka dapat dipakai sendiri, maka multiplisitas dari superclass ke subclass adalah 0..1, jika tidak, itu adalah 1..1. Selanjutnya, batasan eksklusif-atau (XOR) harus ditambahkan di antara asosiasi. Lakukan ini untuk setiap superclass.
atau
Aturan 8b: Ratakan hierarki pewarisan dengan menyalin atribut superclass ke semua subclass dan hapus superclass dari desain.*

*Adalah juga ide yang baik untuk mendokumentasikan modifikasi ini dalam desain sehingga di masa depan, modifikasi pada desain dapat dipertahankan dengan mudah.

Gambar 9-9 Skema Pemetaan Objek Domain Masalah ke RDBMS



Gambar 9-10 Contoh Pemetaan Objek Domain Masalah ke Skema RDBMS

Aturan kelima membahas atribut multinilai dan kelompok atribut berulang dalam objek domain masalah. Dalam kasus ini, atribut harus digunakan untuk membuat tabel baru di RDBMS. Seperti dalam pemetaan ORDBMS, kelompok atribut yang berulang dapat mengimplikasikan hilangnya kelas di lapisan Domain Masalah. Dalam hal ini, kelas domain masalah baru mungkin diperlukan. Terakhir, kita harus membuat asosiasi satu-ke-banyak atau nol-ke-banyak dari tabel asli ke tabel baru. Sebagai contoh, pada Gambar 9-10, kita perlu membuat tabel baru untuk operator asuransi karena pasien dapat memiliki lebih dari satu operator asuransi.

Aturan keenam mendukung pemetaan hubungan agregasi dan asosiasi multinilai (banyak ke banyak) ke tabel baru yang menghubungkan dua tabel asli. Dalam hal ini, tabel

baru harus berisi utama asing kembali ke tabel asli. Sebagai contoh, pada Gambar 9-10, kita perlu membuat tabel baru yang merepresentasikan hubungan penderitaan antara kelas domain masalah Pasien dan Gejala.

Aturan ketujuh membahas hubungan satu-ke-banyak dan banyak-ke-satu. Dengan jenis hubungan ini, sisi multinilai (0..* atau 1..*) harus dipetakan ke kolom dalam tabelnya yang dapat menyimpan utama asing kembali ke sisi bernilai tunggal (0..1 atau 1. .1). Ada kemungkinan bahwa kami telah menangani situasi ini karena sebelumnya kami merekomendasikan penyertaan atribut utama primer dan utama asing dalam kelas domain masalah. Dalam kasus Gambar 9-10, kami telah menambahkan utama utama dari kelas Pasien ke kelas Appointment sebagai utama asing (lihat nomor peserta). Namun, dalam kasus hubungan refleksif, operator asuransi utama, yang terkait dengan kelas Pasien, kita perlu menambahkan atribut baru (PrimerInsuranceCarrier) untuk dapat menyimpan hubungan tersebut.

Aturan kedelapan, dan terakhir, berkaitan dengan kurangnya dukungan untuk generalisasi dan pewarisan. Seperti dalam kasus ORDBMS, ada dua pendekatan yang berbeda. Pendekatan ini hampir identik dengan aturan yang dijelaskan dengan format persistensi objek OODBMS dan ORDBMS yang diberikan sebelumnya. Pendekatan pertama adalah menambahkan kolom ke setiap tabel yang mewakili subclass untuk setiap superclass konkret dari subclass. Pada dasarnya, ini memastikan bahwa utama utama dari subkelas sama dengan utama utama untuk superkelas. Jika sebelumnya kami telah menambahkan utama utama dan utama asing ke objek domain masalah, seperti yang kami sarankan, maka kami tidak perlu melakukan hal lain. Utama utama tabel akan digunakan untuk bergabung kembali dengan instance yang disimpan dalam tabel yang mewakili setiap bagian dari objek domain masalah. Sebaliknya, hierarki pewarisan dapat diratakan dan aturan (Aturan 1 hingga 7) dapat diterapkan kembali.

Seperti dalam kasus pendekatan ORDBMS, pemrosesan tambahan akan diperlukan setiap kali terjadi interaksi antara database dan sistem. Setiap kali sebuah objek harus dibuat, diambil dari database, diperbarui, atau dihapus, pemetaan antara domain masalah dan RDBMS harus digunakan untuk mengkonversi antara dua format yang berbeda. Dalam hal ini, banyak pemrosesan tambahan akan diperlukan.

9.5 MENGOPTIMALKAN PENYIMPANAN OBYEK BERBASIS RDBMSMS

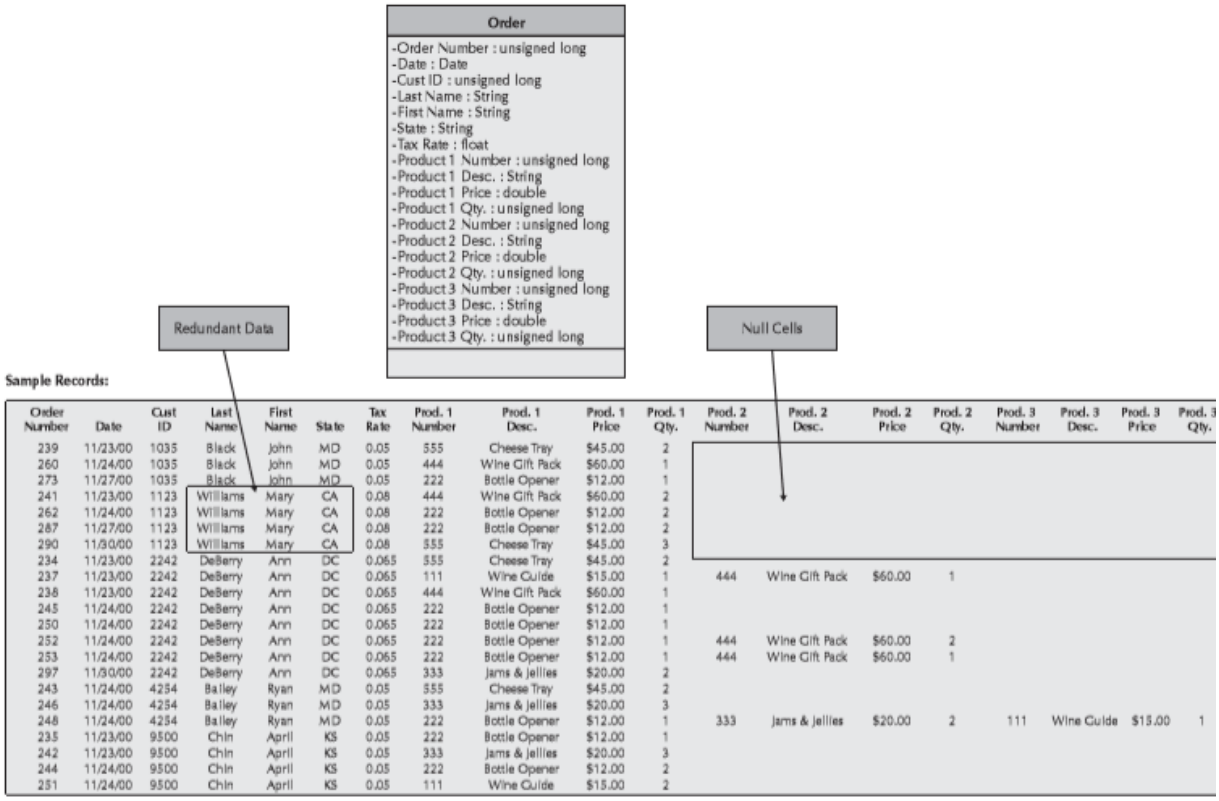
Setelah format persistensi objek dipilih, langkah kedua adalah mengoptimalkan persistensi objek untuk efisiensi pemrosesan. Metode pengoptimalan bervariasi berdasarkan format yang Anda pilih; namun, konsep dasarnya tetap sama. Setelah Anda memahami cara mengoptimalkan jenis persistensi objek tertentu, Anda akan memiliki beberapa gagasan tentang cara mendekati pengoptimalan format lain. Bagian ini berfokus pada optimalisasi format penyimpanan paling populer: database relasional.

Ada dua dimensi utama untuk mengoptimalkan database relasional: untuk efisiensi penyimpanan dan untuk kecepatan akses. Sayangnya, kedua tujuan ini sering bertentangan karena desain terbaik untuk kecepatan akses mungkin memakan banyak ruang penyimpanan dibandingkan dengan desain lain yang kurang cepat. Pada akhirnya, tim proyek akan melalui serangkaian trade-off sampai keseimbangan yang ideal tercapai.

Mengoptimalkan Efisiensi Penyimpanan

Tabel yang paling efisien dalam database relasional dalam hal ruang penyimpanan tidak memiliki data yang berlebihan dan sangat sedikit nilai nol. Kehadiran nilai nol menunjukkan bahwa ruang yang terbuang (dan lebih banyak data untuk disimpan berarti biaya

perangkat keras penyimpanan data yang lebih tinggi). Misalnya, tabel pada Gambar 9-11 mengulangi informasi pelanggan, seperti nama dan negara bagian, setiap kali pelanggan melakukan pemesanan, dan tabel tersebut berisi banyak nilai nol di kolom terkait produk. Nol ini terjadi setiap kali pelanggan memesan kurang dari tiga item (jumlah maksimum pada pesanan).



Order Number	Date	Cust ID	Last Name	First Name	State	Tax Rate	Prod. 1 Number	Prod. 1 Desc.	Prod. 1 Price	Prod. 1 Qty.	Prod. 2 Number	Prod. 2 Desc.	Prod. 2 Price	Prod. 2 Qty.	Prod. 3 Number	Prod. 3 Desc.	Prod. 3 Price	Prod. 3 Qty.
239	11/23/00	1035	Black	John	MD	0.05	555	Cheese Tray	$45.00	2								
260	11/24/00	1035	Black	John	MD	0.05	444	Wine Gift Pack	$60.00	1								
273	11/27/00	1035	Black	John	MD	0.05	222	Bottle Opener	$12.00	1								
241	11/23/00	1123	Williams	Mary	CA	0.08	444	Wine Gift Pack	$60.00	2								
262	11/24/00	1123	Williams	Mary	CA	0.08	222	Bottle Opener	$12.00	2								
287	11/27/00	1123	Williams	Mary	CA	0.08	222	Bottle Opener	$12.00	2								
290	11/30/00	1123	Williams	Mary	CA	0.08	555	Cheese Tray	$45.00	3								
234	11/23/00	2242	DeBerry	Ann	DC	0.065	555	Cheese Tray	$45.00	2								
237	11/23/00	2242	DeBerry	Ann	DC	0.065	111	Wine Guide	$15.00	1	444	Wine Gift Pack	$60.00	1				
238	11/23/00	2242	DeBerry	Ann	DC	0.065	444	Wine Gift Pack	$60.00	1								
245	11/24/00	2242	DeBerry	Ann	DC	0.065	222	Bottle Opener	$12.00	1								
250	11/24/00	2242	DeBerry	Ann	DC	0.065	222	Bottle Opener	$12.00	1								
252	11/24/00	2242	DeBerry	Ann	DC	0.065	222	Bottle Opener	$12.00	1	444	Wine Gift Pack	$60.00	2				
253	11/24/00	2242	DeBerry	Ann	DC	0.065	222	Bottle Opener	$12.00	1	444	Wine Gift Pack	$60.00	1				
297	11/30/00	2242	DeBerry	Ann	DC	0.065	333	Jams & Jellies	$20.00	2								
243	11/24/00	4254	Bailey	Ryan	MD	0.05	555	Cheese Tray	$45.00	2								
246	11/24/00	4254	Bailey	Ryan	MD	0.05	333	Jams & Jellies	$20.00	3								
248	11/24/00	4254	Bailey	Ryan	MD	0.05	222	Bottle Opener	$12.00	1	333	Jams & Jellies	$20.00	2	111	Wine Guide	$15.00	1
235	11/23/00	9500	Chin	April	KS	0.05	222	Bottle Opener	$12.00	1								
242	11/23/00	9500	Chin	April	KS	0.05	333	Jams & Jellies	$20.00	3								
244	11/24/00	9500	Chin	April	KS	0.05	222	Bottle Opener	$12.00	2								
251	11/24/00	9500	Chin	April	KS	0.05	111	Wine Guide	$15.00	2								

Gambar 9-11 Optimisasi Penyimpanan

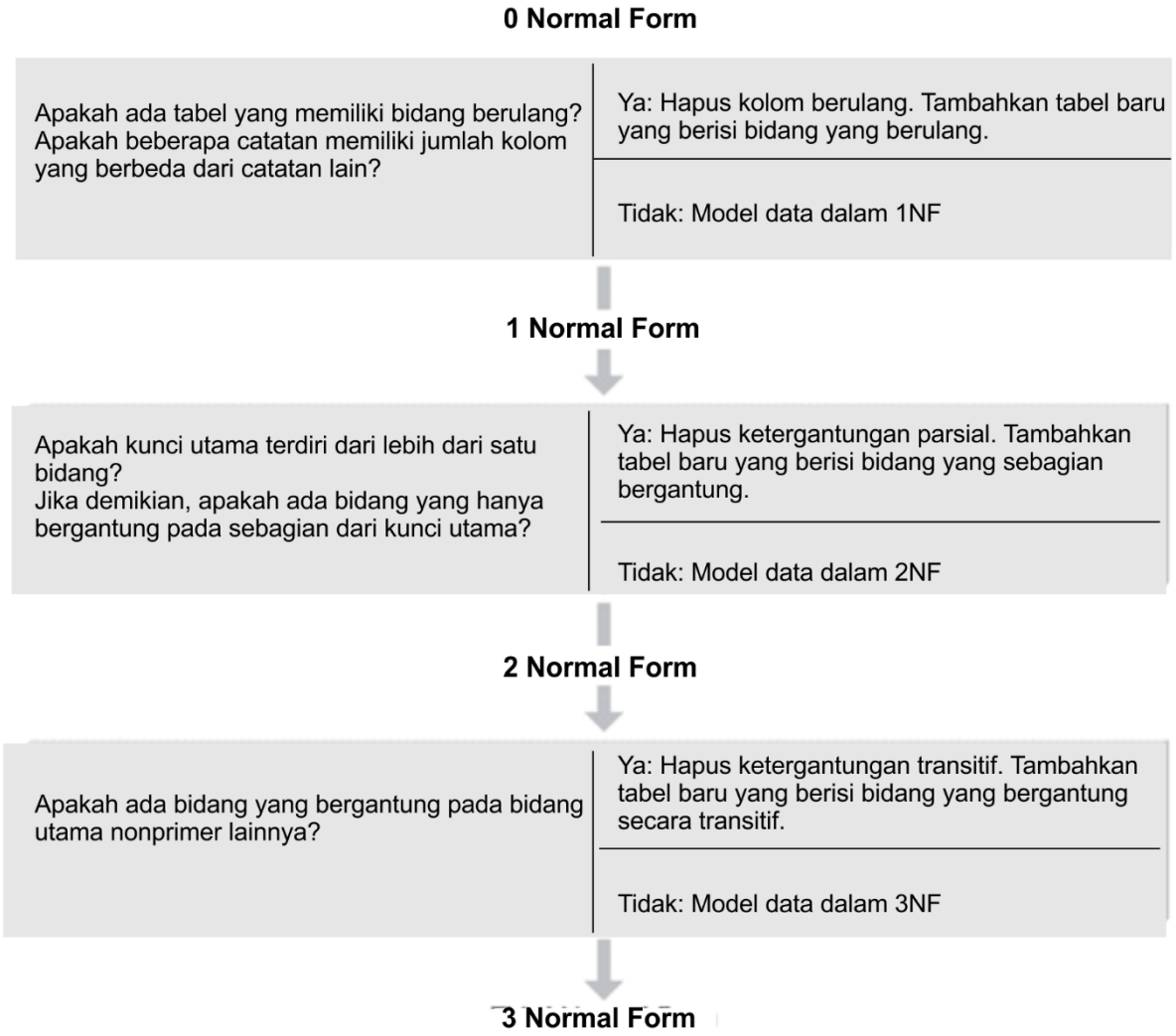
Selain membuang-buang ruang, redundansi dan nilai nol juga memungkinkan lebih banyak ruang untuk kesalahan dan meningkatkan kemungkinan masalah akan muncul dengan integritas data. Bagaimana jika pelanggan 1035 pindah dari Maryland ke Georgia? Dalam kasus Gambar 9-11, sebuah program harus ditulis untuk memastikan bahwa semua contoh pelanggan tersebut diperbarui untuk menunjukkan Georgia sebagai negara bagian tempat tinggal yang baru. Jika beberapa contoh diabaikan, maka tabel akan berisi anomali pembaruan, di mana beberapa catatan berisi nilai status yang diperbarui dengan benar dan catatan lain berisi informasi lama.

Nulls mengancam integritas data karena sulit untuk ditafsirkan. Nilai kosong pada kolom produk tabel Pesanan dapat berarti pelanggan tidak menginginkan lebih dari satu atau dua produk pada pesanannya, operator lupa memasukkan ketiga produk pada pesanan, atau pelanggan membatalkan sebagian pesanan dan produk telah dihapus oleh operator. Tidak mungkin untuk memastikan arti sebenarnya dari nol.

Untuk kedua alasan ini—ruang penyimpanan yang terbuang dan ancaman integritas data—tim proyek harus menghapus redundansi dan null dari tabel. Selama desain, diagram kelas digunakan untuk memeriksa desain tabel RDBMS (misalnya, lihat Gambar 9-10) dan untuk mengoptimalkannya untuk efisiensi penyimpanan. Jika Anda mengikuti instruksi dan panduan pemodelan yang disajikan di Bab 5, Anda akan mengalami sedikit kesulitan membuat desain yang sangat dioptimalkan dengan cara ini karena model data logis yang terbentuk dengan baik tidak mengandung redundansi atau banyak nilai nol.

Namun, terkadang, tim proyek perlu memulai dengan model yang dibuat dengan buruk atau dengan model yang dibuat untuk file atau tipe format nonrelasional. Dalam kasus ini, tim proyek harus mengikuti serangkaian langkah yang berfungsi untuk memeriksa model untuk efisiensi penyimpanan. Langkah-langkah ini membentuk proses yang disebut normalisasi. Normalisasi adalah proses di mana serangkaian aturan diterapkan ke tabel RDBMS untuk menilai efisiensi tabel (lihat Gambar 9-12). Aturan ini membantu analis mengidentifikasi tabel yang tidak direpresentasikan dengan benar. Di sini, kami menjelaskan tiga aturan normalisasi yang diterapkan secara teratur dalam praktik. Gambar 9-11 menunjukkan model dalam 0 Bentuk Normal, yang merupakan model yang tidak dinormalisasi sebelum aturan normalisasi diterapkan.

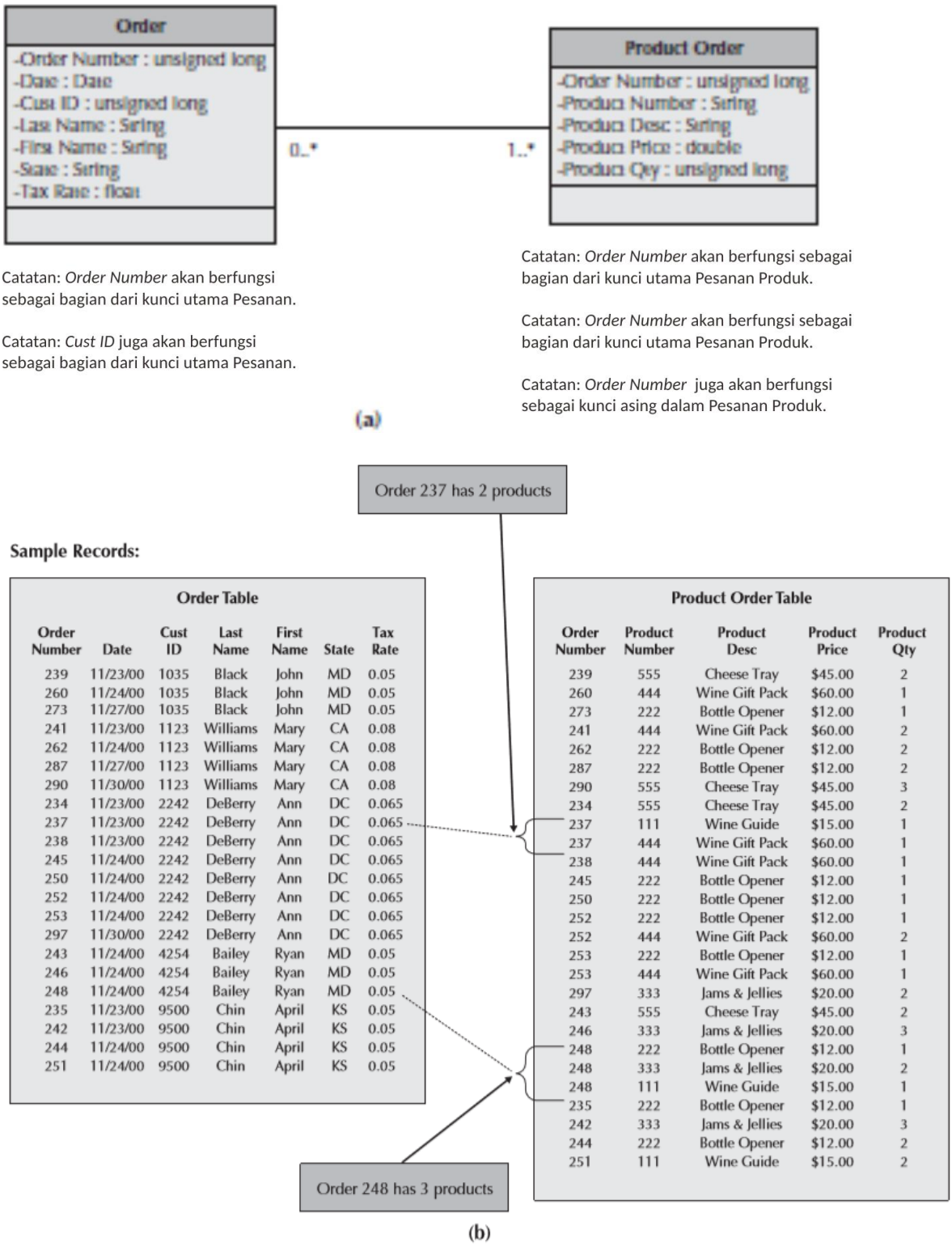
Sebuah model berada dalam bentuk normal pertama (1NF) jika tidak mengarah ke bidang multinilai, bidang yang memungkinkan sekumpulan nilai untuk disimpan, atau bidang berulang, yaitu bidang yang berulang dalam tabel untuk menangkap beberapa nilai. Aturan untuk 1NF mengatakan bahwa semua tabel harus berisi jumlah kolom yang sama (yaitu, bidang) dan bahwa semua kolom harus berisi satu nilai. Perhatikan bahwa model pada Gambar 9-11 melanggar 1NF karena menyebabkan nomor produk, deskripsi, harga, dan kuantitas berulang tiga kali untuk setiap pesanan dalam tabel. Tabel yang dihasilkan memiliki banyak catatan yang berisi nol di kolom terkait produk, dan pesanan dibatasi hingga tiga produk karena tidak ada ruang untuk menyimpan informasi lebih lanjut.



Gambar 9-12 Langkah-Langkah Normalisasi

Desain yang jauh lebih efisien (dan yang sesuai dengan 1NF) mengarah ke tabel terpisah untuk menyimpan informasi berulang; untuk melakukan ini, kami membuat tabel terpisah pada model untuk menangkap informasi pesanan produk. Hubungan nol-ke-banyak kemudian akan ada di antara dua tabel. Seperti yang ditunjukkan pada Gambar 9-13, desain baru menghilangkan nol dari tabel Pesanan dan mendukung jumlah produk yang tidak terbatas yang dapat dikaitkan dengan pesanan.

Model yang direvisi:



Gambar 9-13 1NF: Hapus Kolom Berulang

Bentuk normal kedua (2NF) membutuhkan pertama bahwa model data dalam 1NF dan kedua bahwa model data mengarah ke tabel yang berisi bidang yang bergantung pada seluruh utama utama. Ini berarti bahwa nilai utama utama untuk setiap catatan dapat menentukan nilai untuk semua bidang lain dalam catatan. Terkadang bidang hanya bergantung pada sebagian utama utama (yaitu, ketergantungan sebagian), dan bidang ini termasuk dalam tabel lain.

Misalnya, dalam tabel Pesanan Produk baru yang dibuat pada Gambar 9-13, utama utama adalah kombinasi nomor pesanan dan nomor produk, tetapi deskripsi produk dan atribut harga hanya bergantung pada nomor produk. Dengan kata lain, dengan mengetahui nomor produk, kita dapat mengidentifikasi deskripsi dan harga produk. Namun, pengetahuan tentang nomor pesanan dan nomor produk diperlukan untuk mengidentifikasi kuantitas. Untuk memperbaiki pelanggaran 2NF ini, tabel dibuat untuk menyimpan informasi produk, dan atribut deskripsi dan harga dipindahkan ke tabel baru. Sekarang, deskripsi produk disimpan hanya sekali untuk setiap nomor produk, bukan berkali-kali (setiap kali produk dipesan).

Pelanggaran kedua 2NF terjadi di tabel Pesanan: nama depan dan nama belakang pelanggan hanya bergantung pada ID pelanggan, bukan seluruh utama (ID Pelanggan dan nomor Pesanan). Akibatnya, setiap kali ID pelanggan muncul di tabel Pesanan, nama juga muncul. Cara yang jauh lebih ekonomis untuk menyimpan data adalah dengan membuat tabel Pelanggan dengan ID Pelanggan sebagai utama utama dan bidang terkait pelanggan lainnya (yaitu, nama belakang dan nama depan) terdaftar hanya sekali dalam catatan yang sesuai. Gambar 9-14 mengilustrasikan bagaimana model akan terlihat ketika ditempatkan di 2NF.

Bentuk normal ketiga (3NF) terjadi ketika model berada dalam 1NF dan 2NF dan, dalam tabel yang dihasilkan, tidak ada bidang yang bergantung pada bidang utama nonprimer (yaitu, ketergantungan transitif). Gambar 9-14 berisi pelanggaran 3NF: Tarif pajak atas pesanan tergantung pada negara bagian tujuan pengiriman pesanan. Solusinya melibatkan pembuatan tabel lain yang berisi singkatan negara bagian yang berfungsi sebagai utama utama dan tarif pajak sebagai bidang reguler. Gambar 9-15 menyajikan hasil akhir penerapan langkah-langkah normalisasi ke model asli dari Gambar 9-11.

Mengoptimalkan Kecepatan Akses Data

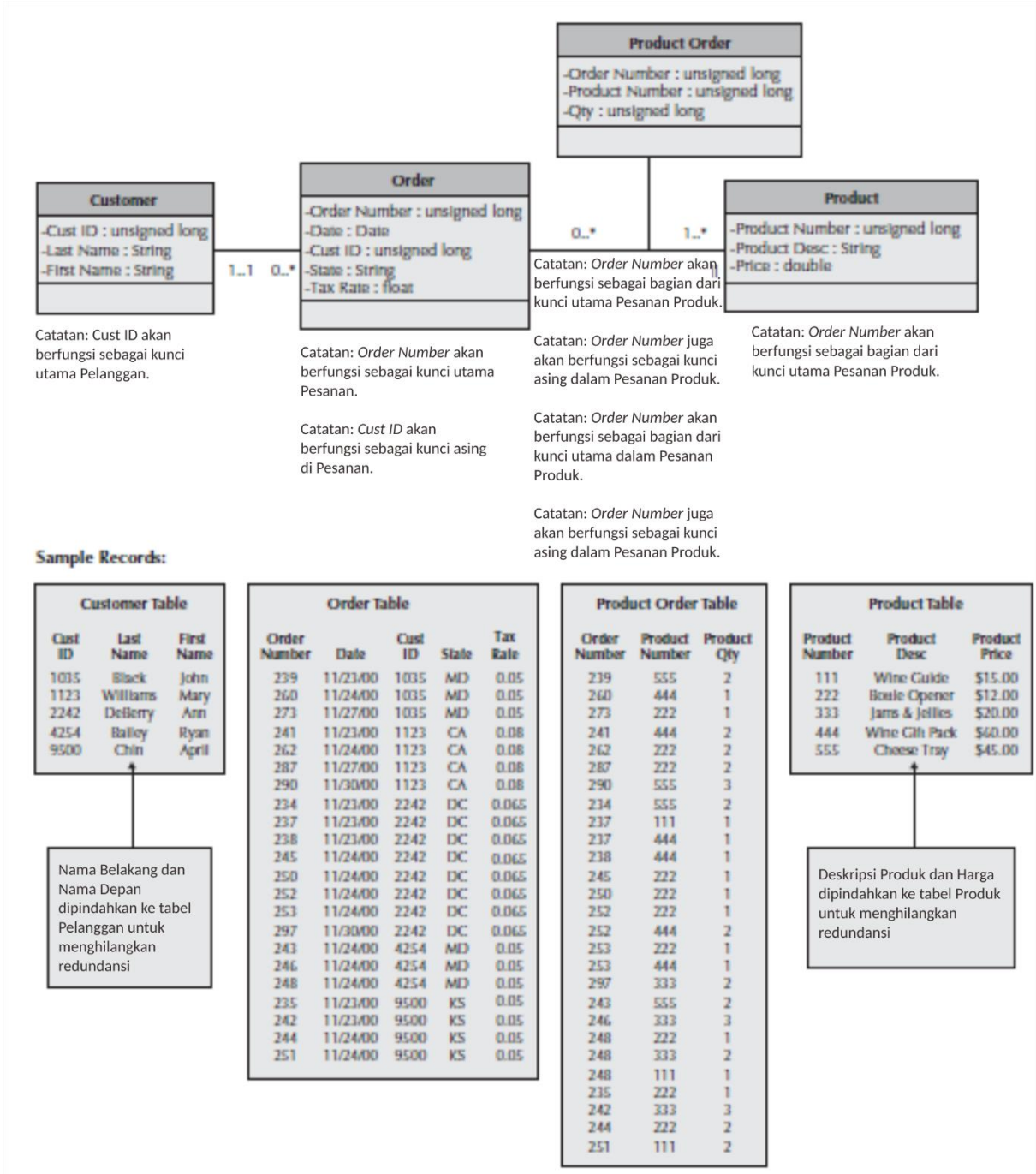
Setelah Anda mengoptimalkan desain penyimpanan objek untuk efisiensi, hasil akhirnya adalah data tersebar di sejumlah tabel. Ketika data dari beberapa tabel perlu diakses atau di-query, tabel-tabel tersebut harus digabungkan terlebih dahulu. Misalnya, sebelum pengguna dapat mencetak daftar nama pelanggan yang terkait dengan pesanan, terlebih dahulu tabel Pelanggan dan Pesanan harus digabungkan, berdasarkan bidang nomor pelanggan (lihat Gambar 9-15). Hanya dengan begitu informasi pesanan dan pelanggan dapat disertakan dalam output kueri. Penggabungan dapat memakan banyak waktu, terutama jika tabelnya besar atau jika banyak tabel yang terlibat.

Pertimbangkan sebuah sistem yang menyimpan informasi tentang 10.000 produk berbeda, 25.000 pelanggan, dan 100.000 pesanan, masing-masing dengan rata-rata tiga produk per pesanan. Jika seorang analis ingin menyelidiki apakah ada perbedaan regional dalam preferensi musik, dia perlu menggabungkan semua tabel untuk dapat melihat produk yang telah dipesan sambil mengetahui keadaan pelanggan yang melakukan pemesanan. Permintaan informasi ini akan menghasilkan tabel besar dengan 300.000 baris (yaitu, jumlah produk yang telah dipesan) dan 11 kolom (jumlah total kolom dari semua tabel digabungkan).

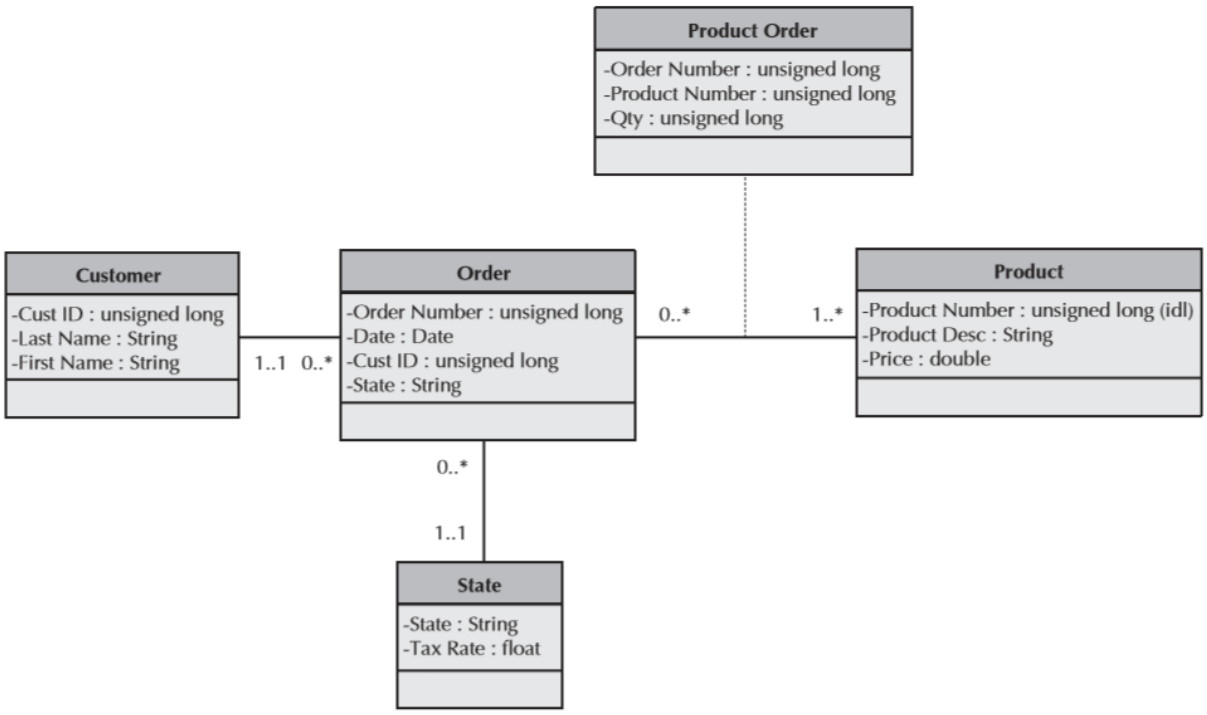
Tim proyek dapat menggunakan beberapa teknik untuk mencoba mempercepat akses ke data, termasuk denormalisasi, pengelompokan, dan pengindeksan.

Denormalisasi Setelah penyimpanan objek dioptimalkan, tim proyek dapat memutuskan bahwa peningkatan kecepatan pengambilan data lebih penting daripada efisiensi penyimpanan atau kecepatan pembaruan data dan memilih untuk mendenormalisasi atau menambahkan redundansi kembali ke dalam desain. Denormalisasi mengurangi jumlah gabungan yang perlu dilakukan dalam kueri, sehingga mempercepat akses. Gambar 9-16 menunjukkan model denormalisasi untuk pesanan pelanggan. Nama belakang pelanggan ditambahkan kembali ke tabel Pesanan karena tim proyek belajar selama analisis bahwa kueri tentang pesanan biasanya memerlukan bidang nama belakang pelanggan. Alih-alih menggabungkan tabel Pesanan berulang kali ke tabel Pelanggan, sistem sekarang hanya perlu mengakses tabel Pesanan karena berisi semua informasi relevan yang diperlukan untuk menyelesaikan pertanyaan preferensi musik yang diajukan di atas.

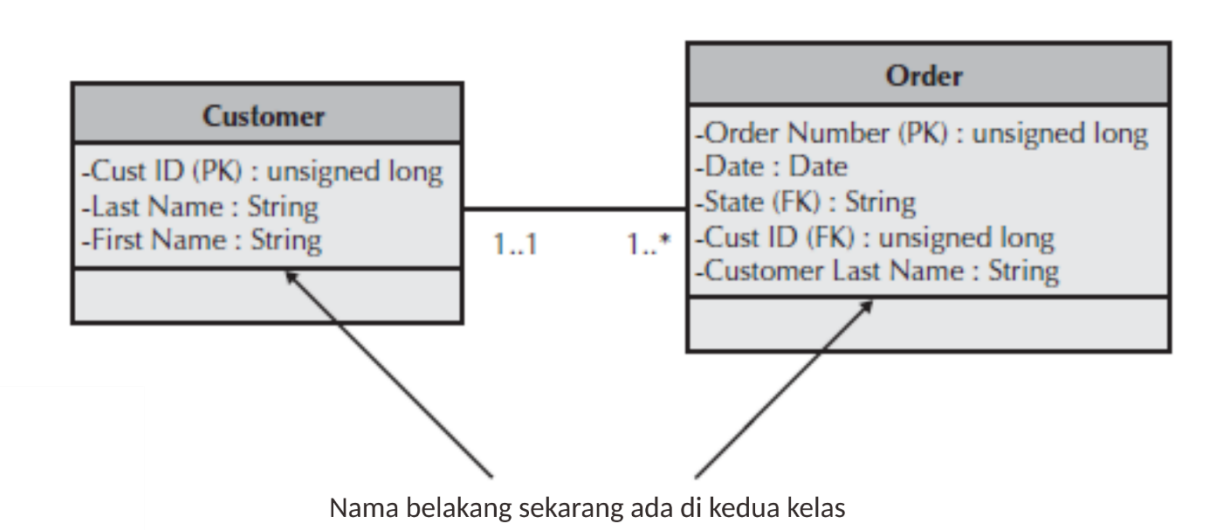
Denormalisasi sangat ideal dalam situasi di mana informasi sering ditanyakan tetapi jarang diperbarui. Namun, karena penyimpanan tambahan yang diperlukan dan potensi anomali pembaruan, denormalisasi harus diterapkan dengan hemat. Ada tiga kasus di mana Anda dapat mengandalkan denormalisasi untuk mengurangi gabungan dan meningkatkan kinerja. Pertama, denormalisasi dapat diterapkan dalam kasus tabel pencarian, yaitu tabel yang berisi deskripsi nilai (misalnya, tabel deskripsi produk atau tabel jenis pembayaran). Karena deskripsi kode jarang berubah, mungkin lebih efisien untuk memasukkan deskripsi bersama dengan kode masing-masing di tabel utama untuk menghilangkan kebutuhan untuk bergabung dengan tabel pencarian setiap kali kueri dilakukan (lihat Gambar 9-17a).



Gambar 9-14 2NF Ketergantungan Parsial Dihapus



Gambar 9-15 3NF Bidang Normalisasi



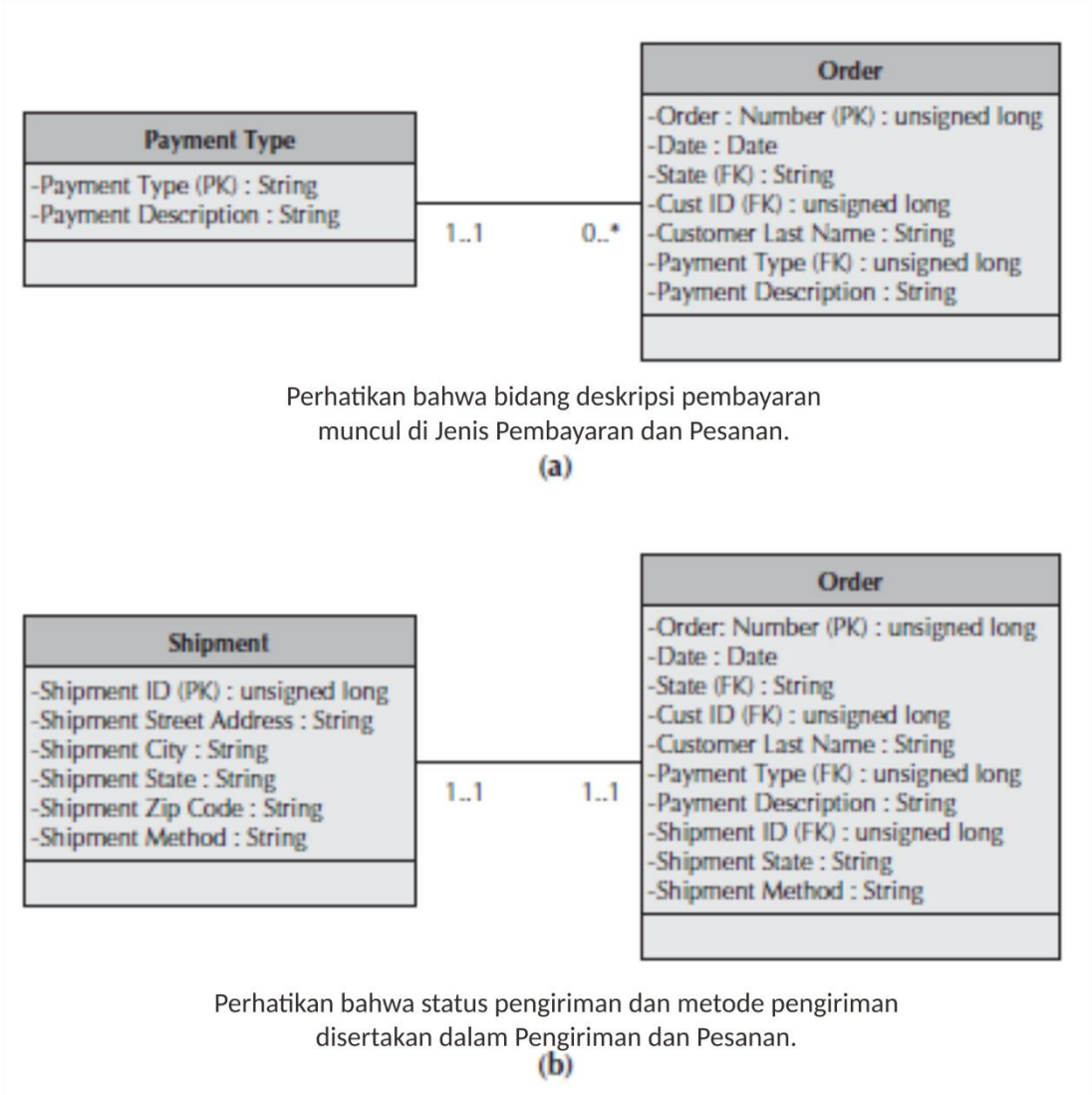
Gambar 9-16 Model Data Fisik yang Didenormalisasi

Kedua, hubungan satu-ke-satu adalah kandidat yang baik untuk denormalisasi. Meskipun secara logis dua tabel harus dipisahkan, dari sudut pandang praktis informasi dari kedua tabel dapat diakses secara teratur bersama-sama. Pikirkan tentang pesanan dan informasi pengirimannya. Logikanya, mungkin masuk akal untuk memisahkan atribut yang terkait dengan pengiriman ke dalam tabel terpisah, tetapi sebagai hasilnya, kueri terkait pengiriman mungkin akan selalu memerlukan gabungan ke tabel Pesanan. Jika tim proyek menemukan bahwa informasi pengiriman tertentu, seperti status dan metode pengiriman, diperlukan saat pesanan diakses, mereka dapat memutuskan untuk menggabungkan tabel atau menyertakan beberapa atribut pengiriman dalam tabel Pesanan (lihat Gambar 9-17b).

Ketiga, terkadang lebih efisien untuk menyertakan atribut entitas induk dalam entitas anaknya pada model data fisik. Sebagai contoh, perhatikan tabel Pelanggan dan Pesanan pada

Gambar 9-16, yang berbagi hubungan satu-ke-banyak, dengan Pelanggan sebagai induk dan Pesanan sebagai anak.

Jika pertanyaan mengenai pesanan terus menerus memerlukan informasi pelanggan, bidang pelanggan yang paling populer dapat ditempatkan di Order untuk mengurangi penggabungan yang diperlukan ke tabel Pelanggan, seperti yang dilakukan dengan Nama Belakang Pelanggan.



Gambar 9-17 Situasi Denormalisasi (FK, foreign key; PK, primary key)

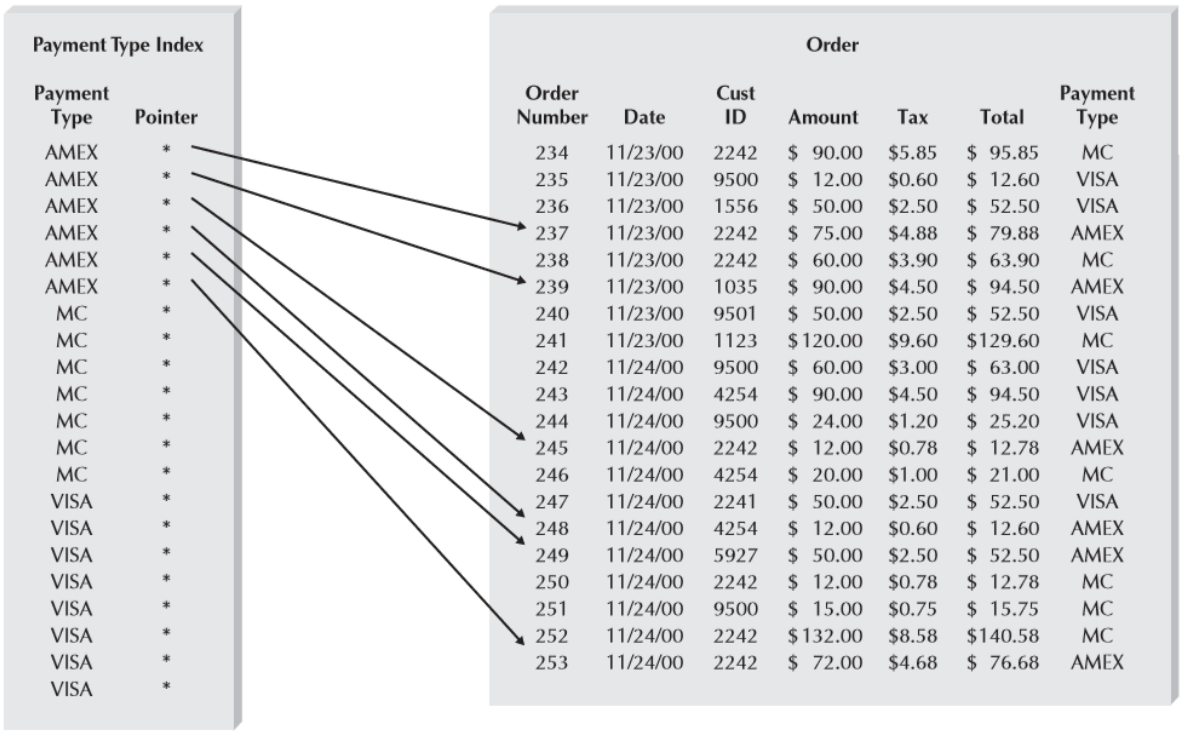
Clustering Kecepatan akses juga dipengaruhi oleh cara data diambil. Pikirkan tentang berbelanja di toko kelontong. Jika Anda memiliki daftar barang untuk dibeli tetapi Anda tidak terbiasa dengan tata letak toko, Anda harus berjalan menyusuri setiap lorong untuk memastikan bahwa Anda tidak melewatkan apa pun dari daftar Anda. Demikian juga, jika catatan disusun tanpa urutan tertentu (atau dalam urutan yang tidak relevan dengan kebutuhan data Anda), maka setiap kueri catatan menghasilkan pemindaian tabel di mana DBMS harus mengakses setiap baris dalam tabel sebelum mengambilnya. hasil yang ditetapkan. Pemindaian tabel adalah metode pengambilan data yang paling tidak efisien.

Salah satu cara untuk meningkatkan kecepatan akses adalah dengan mengurangi berapa kali media penyimpanan perlu diakses selama transaksi. Salah satu metodenya adalah dengan mengelompokkan record secara fisik sehingga record yang serupa disimpan

berdekatan. Dengan pengelompokan intrafile, seperti catatan dalam tabel disimpan bersama dalam beberapa cara, seperti diurutkan berdasarkan utama utama atau, dalam kasus toko kelontong, menurut jenis barang. Jadi, setiap kali sebuah query mencari record, query tersebut dapat langsung menuju ke tempat yang tepat pada disk (atau media penyimpanan lainnya) karena tahu dalam urutan apa record disimpan, sama seperti kita dapat berjalan langsung ke toko roti untuk mengambilnya. sepotong roti. Pengelompokan interfile menggabungkan catatan dari lebih dari satu tabel yang biasanya diambil bersama-sama. Misalnya, jika informasi pelanggan biasanya diakses dengan informasi pesanan terkait, maka catatan dari dua tabel dapat disimpan secara fisik dengan cara yang mempertahankan hubungan pesanan pelanggan. Kembali ke skenario toko kelontong, cluster interfile akan mirip dengan menyimpan selai kacang, jeli, dan roti di samping satu sama lain di lorong yang sama karena biasanya dibeli bersama, bukan karena jenis barang yang serupa. Tentu saja, setiap tabel hanya dapat memiliki satu strategi pengelompokan karena catatan dapat diatur secara fisik hanya dalam satu cara.

Pengindeksan. Penghemat waktu yang familier adalah indeks yang terletak di bagian belakang buku teks, yang menunjuk langsung ke halaman atau halaman yang berisi topik yang menarik. Pikirkan berapa lama waktu yang dibutuhkan untuk menemukan semua waktu yang database relasional muncul di buku teks ini tanpa indeks untuk diandalkan! Indeks dalam penyimpanan data seperti indeks di belakang buku teks; itu adalah minitable yang berisi nilai dari satu atau lebih kolom dalam tabel dan lokasi nilai di dalam tabel. Alih-alih membuka halaman melalui seluruh buku teks, kita dapat langsung berpindah ke halaman yang tepat dan mendapatkan informasi yang kita butuhkan. Indeks adalah salah satu cara terpenting untuk meningkatkan kinerja database. Setiap kali ada masalah kinerja, tempat pertama yang harus dilihat adalah indeks.

Kueri dapat menggunakan indeks untuk menemukan lokasi hanya rekaman yang disertakan dalam jawaban kueri, dan tabel dapat memiliki jumlah indeks yang tidak terbatas. Gambar 9-18 menunjukkan indeks yang mencatat pesanan berdasarkan jenis pembayaran. Kueri yang mencari semua pelanggan yang menggunakan American Express dapat menggunakan indeks ini untuk menemukan lokasi catatan yang berisi American Express sebagai jenis pembayaran tanpa harus memindai seluruh tabel Pesanan.



Gambar 9-18 Indeks Jenis Pembayaran

Gunakan indeks hemat untuk sistem transaksi. Gunakan banyak indeks untuk meningkatkan waktu respons dalam sistem pendukung keputusan. Untuk setiap tabel, buat indeks unik yang didasarkan pada utama utama. Untuk setiap tabel, buat indeks yang didasarkan pada utama asing untuk meningkatkan kinerja gabungan. Buat indeks untuk bidang yang sering digunakan untuk pengelompokan, pengurutan, atau kriteria.

Gambar 9-19 Pedoman Membuat Indeks

Tim proyek dapat membuat indeks bekerja lebih cepat dengan menempatkannya ke dalam memori utama perangkat keras penyimpanan data. Mengambil informasi langsung dari memori jauh lebih cepat daripada mengambilnya dari hard disk—Pikirkan tentang seberapa cepat mengambil nomor telepon yang diingat dibandingkan dengan nomor yang harus dicari di buku telepon. Demikian pula, ketika database memiliki indeks di memori, database dapat menemukan catatan dengan sangat, sangat cepat.

Tentu saja, indeks memerlukan overhead karena mereka mengambil ruang pada media penyimpanan. Juga, mereka perlu diperbarui saat catatan dalam tabel dimasukkan, dihapus, atau diubah. Jadi, meskipun indeks mengarah pada akses yang lebih cepat ke data, mereka memperlambat proses pembaruan. Secara umum, kita harus membuat indeks dengan hemat untuk sistem transaksi atau sistem yang membutuhkan banyak pembaruan, tetapi kita harus menerapkan indeks dengan murah hati saat merancang sistem untuk pendukung keputusan (lihat Gambar 9-19).

Memperkirakan Ukuran Penyimpanan Data

Bahkan jika kami telah mendenormalisasi model data fisik kami, catatan berkerumun, dan membuat indeks dengan tepat, sistem akan berkinerja buruk jika server database tidak dapat menangani volume datanya. Oleh karena itu, satu cara terakhir untuk merencanakan kinerja yang baik adalah dengan menerapkan volumetrik, yang berarti memperkirakan jumlah data yang perlu didukung oleh perangkat keras. Anda dapat memasukkan perkiraan Anda ke dalam spesifikasi perangkat keras server database untuk memastikan bahwa perangkat keras database cukup untuk kebutuhan proyek. Ukuran database didasarkan pada jumlah data mentah dalam tabel dan kebutuhan overhead DBMS. Untuk memperkirakan ukuran, Anda harus memiliki pemahaman yang baik tentang ukuran awal database Anda serta tingkat pertumbuhan yang diharapkan dari waktu ke waktu.

Data mentah mengacu pada semua data yang disimpan dalam tabel database, dan dihitung berdasarkan pendekatan bottom-up. Pertama, tuliskan perkiraan lebar rata-rata untuk setiap kolom (bidang) dalam tabel dan jumlahkan nilainya untuk ukuran catatan total (lihat Gambar 9-20). Misalnya, jika kolom Nama Belakang dengan lebar variabel diberi lebar 20 karakter, Anda dapat memasukkan 13 sebagai lebar karakter rata-rata kolom. Pada Gambar 9-20, perkiraan ukuran record adalah 49.

Selanjutnya, hitung overhead untuk tabel sebagai persentase dari setiap record. Overhead mencakup ruangan yang dibutuhkan oleh DBMS untuk mendukung fungsi seperti tindakan administratif dan indeks, dan harus ditetapkan berdasarkan pengalaman masa lalu, rekomendasi dari vendor teknologi, atau parameter yang dibangun ke dalam perangkat lunak yang ditulis untuk menghitung volumetrik. Misalnya, vendor DBMS Anda mungkin menyarankan Anda mengalokasikan 30 persen dari ukuran data mentah record untuk ruang penyimpanan overhead, membuat total ukuran record 63,7 pada contoh Gambar 9-20.

Terakhir, catat jumlah record awal yang akan dimuat ke dalam tabel, serta perkiraan pertumbuhan per bulan. Informasi ini seharusnya dikumpulkan selama analisis. Menurut Gambar 9-20, ruang awal yang dibutuhkan oleh tabel pertama adalah 3.185.000, dan ukuran masa depan dapat diproyeksikan berdasarkan angka pertumbuhan. Langkah-langkah ini diulang untuk setiap tabel untuk mendapatkan ukuran total untuk seluruh database.

Field	Average Size
Order Number	8
Date	7
Cust ID	4
Last Name	13
First Name	9
State	2
Amount	4
Tax Rate	2
Record Size	49
Overhead	30%
Total Record Size	63.7
Initial Table Size	50,000
Initial Table Volume	3,185,000
Growth Rate/Month	1,000
Table Volume @ 3 years	5,478,200

Gambar 9-20 Menghitung Volumetrik

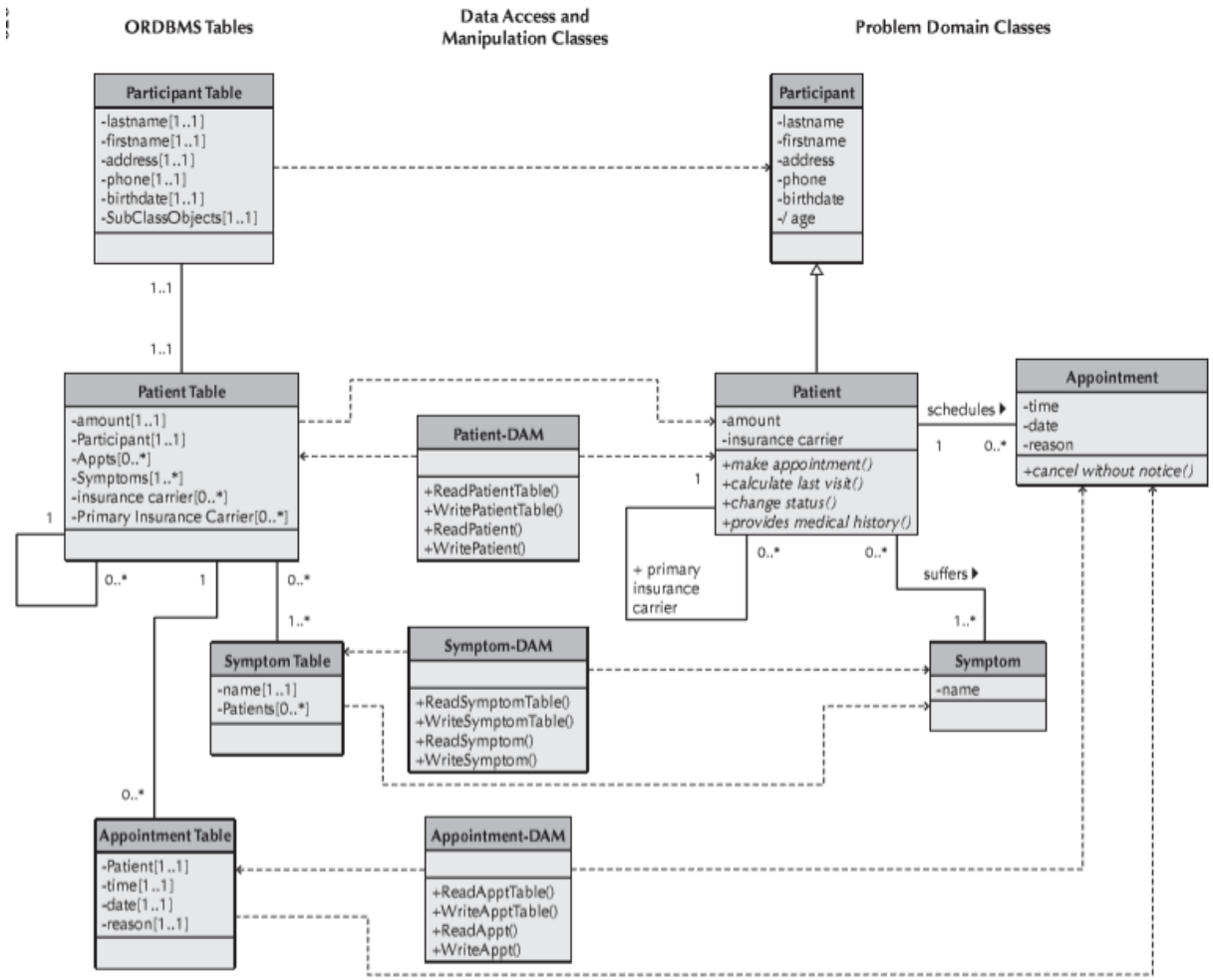
Banyak CASE tool memberi Anda informasi ukuran database berdasarkan cara Anda menyiapkan kegigihan objek, dan alat tersebut menghitung perkiraan volumetrik secara otomatis. Pada akhirnya, ukuran database perlu dibagikan dengan tim desain sehingga teknologi yang tepat dapat diterapkan untuk mendukung data sistem dan potensi masalah kinerja dapat diatasi jauh sebelum mempengaruhi keberhasilan sistem.

9.6 MERANCANG KELAS AKSES DATA DAN MANIPULASI

Langkah terakhir dalam mengembangkan lapisan manajemen data adalah merancang kelas akses dan manipulasi data yang bertindak sebagai penerjemah antara persistensi objek dan objek domain masalah. Dengan demikian, mereka harus selalu mampu setidaknya membaca dan menulis baik objek persistensi maupun objek domain masalah. Seperti dijelaskan sebelumnya dan di Bab 8, kelas persistensi objek diturunkan dari kelas domain masalah konkret, sedangkan kelas akses dan manipulasi data bergantung pada kelas kegigihan objek dan domain masalah.

Tergantung pada aplikasinya, aturan sederhana yang harus diikuti adalah harus ada satu kelas akses dan manipulasi data untuk setiap kelas domain masalah konkret. Dalam beberapa kasus, mungkin masuk akal untuk membuat kelas akses dan manipulasi data yang terkait dengan kelas interaksi manusia-komputer (lihat Bab 10). Namun, ini menciptakan ketergantungan dari lapisan manajemen data ke lapisan interaksi manusia-komputer. Menambahkan kompleksitas tambahan ini ke desain sistem biasanya tidak disarankan.

Kembali ke solusi ORDBMS untuk contoh sistem Pengangkatan (lihat Gambar 9-8), kita melihat bahwa kita memiliki empat kelas domain masalah dan empat tabel ORDBMS. Mengikuti aturan sebelumnya, kelas DAM agak sederhana. Mereka harus mendukung hanya terjemahan satu-ke-satu antara kelas domain masalah konkret dan tabel ORDBMS (lihat Gambar 9-21). Karena kelas domain masalah Peserta adalah kelas abstrak, hanya tiga kelas akses dan manipulasi data yang diperlukan: Patient-DAM, SymptomDAM, dan Appointment-DAM. Namun, proses untuk membuat turunan dari kelas domain masalah Pasien bisa sangat rumit. Kelas Patient-DAM mungkin harus dapat mengambil informasi dari keempat tabel ORDBMS. Untuk mencapai ini, kelas PatientDAM mengambil informasi dari tabel Pasien. Menggunakan Object-ID yang disimpan dalam nilai atribut yang terkait dengan atribut Participant, Appts, dan Gejala, sisa informasi yang diperlukan untuk membuat instance Patient dapat dengan mudah diambil oleh kelas Patient-DAM.

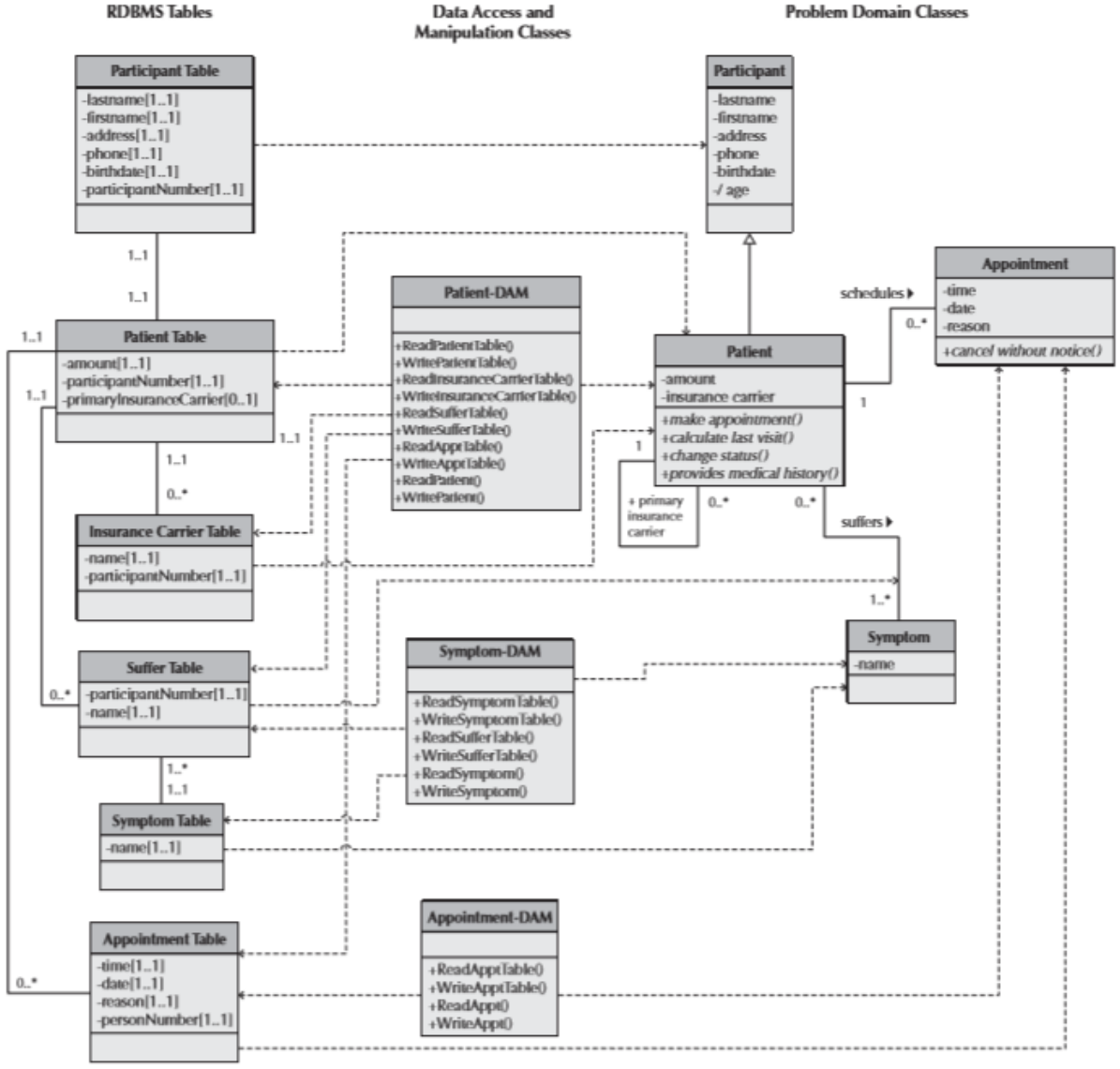


Gambar 9-21 Mengelola Objek Domain Masalah ke ORDBMS Menggunakan Kelas DAM

Dalam Use-Case RDBMS untuk memberikan persistensi, akses data dan kelas manipulasi cenderung menjadi lebih kompleks. Misalnya, dalam sistem Appointment, masih ada empat kelas domain masalah, tetapi karena keterbatasan RDBMS, kita harus mendukung enam tabel RDBMS (lihat Gambar 9-10). Kelas akses dan manipulasi data untuk kelas domain masalah Appointment dan tabel RDBMS Appointment tidak berbeda dari yang didukung untuk solusi ORDBMS (lihat Gambar 9-21 dan 9-22). Namun, karena atribut dan hubungan multinilai yang terkait dengan kelas domain masalah Pasien dan Gejala, pemetaan ke tabel RDBMS menjadi lebih rumit. Akibatnya, jumlah dependensi dari kelas akses dan manipulasi data (DAM Pasien dan DAM Gejala) ke tabel RDBMS (tabel Pasien, tabel Operator Asuransi, tabel Penderita, dan tabel Gejala) telah meningkat. Selanjutnya, karena kelas domain masalah Pasien dikaitkan dengan tiga kelas domain masalah lainnya, pengambilan sebenarnya dari semua informasi yang diperlukan untuk membuat turunan dari kelas Pasien dapat melibatkan penggabungan informasi dari keenam tabel RDBMS. Untuk mencapai hal ini, kelas Pasien-DAM harus terlebih dahulu mengambil informasi dari tabel Pasien, tabel Operator Asuransi, tabel Penderita, dan tabel Pengangkatan. Karena primary key tabel Patient dan tabel Participant identik, kelas Patient-DAM dapat langsung mengambil informasi dari tabel Participant, atau informasi dapat digabungkan menggunakan atribut partisipanNumber dari dua tabel, yang bertindak sebagai keduanya utama utama dan utama asing. Akhirnya, dengan menggunakan informasi yang terdapat dalam tabel Penderita, informasi dalam tabel Gejala juga dapat diambil. Jelas, semakin jauh kita dapatkan dari representasi kelas domain masalah berorientasi objek, semakin banyak pekerjaan yang harus dilakukan. Namun, seperti dalam kasus contoh ORDBMS, perhatikan bahwa sama sekali tidak ada modifikasi yang dilakukan

pada kelas domain masalah. Oleh karena itu, akses data dan kelas manipulasi lagi telah mencegah fungsionalitas manajemen data merayap ke dalam kelas domain masalah.

Salah satu pendekatan khusus yang disarankan untuk mendukung implementasi kelas manipulasi dan akses data adalah dengan menggunakan pustaka pemetaan relasional objek seperti Hibernate. Hibernate, dikembangkan dalam komunitas JBoss, memungkinkan pemetaan objek yang ditulis dalam Java yang akan disimpan dalam RDBMS. Alih-alih menggunakan bahasa pemrograman berorientasi objek untuk mengimplementasikan akses data dan kelas manipulasi, dengan Hibernate, mereka diimplementasikan dalam file XML yang berisi pemetaan. Seperti pada pendekatan di atas, pemodelan pemetaan dalam file XML mencegah detail akses dan manipulasi data menyelinap ke dalam representasi domain masalah.



Gambar 9-22 Memetakan Objek Domain Masalah ke RDBMS Menggunakan Kelas DAM

9.7 PERSYARATAN NON FUNGSIONAL DAN DESAIN LAPISAN MANAJEMEN DATA

Ingatlah bahwa persyaratan nonfungsional mengacu pada sifat perilaku yang harus dimiliki sistem. Properti ini mencakup masalah yang terkait dengan kinerja, keamanan, kemudahan penggunaan, lingkungan operasional, dan keandalan. Dalam teks ini, kami telah mengelompokkan persyaratan nonfungsional ke dalam empat kategori: persyaratan

operasional, kinerja, keamanan, dan budaya dan politik. Kami menjelaskan masing-masing dalam kaitannya dengan lapisan manajemen data.

Persyaratan operasional untuk lapisan manajemen data mencakup masalah yang berhubungan dengan teknologi yang digunakan untuk mendukung persistensi objek. Namun, pilihan perangkat keras dan sistem operasi membatasi pilihan teknologi dan format persistensi objek yang tersedia. Ini terutama benar ketika Anda mempertimbangkan komputasi seluler. Mengingat terbatasnya memori dan penyimpanan yang tersedia pada perangkat ini, pilihan untuk mendukung persistensi objek menjadi terbatas. Salah satu pilihan yang memungkinkan untuk mendukung persistensi objek yang berfungsi baik di platform berbasis Google Android dan iOS Apple adalah SQLite. SQLite adalah versi ringan dari SQL yang mendukung RDBMS. Namun, ada banyak pendekatan berbeda untuk mendukung persistensi objek yang lebih bergantung pada platform; misalnya, Android mendukung penyimpanan objek dengan preferensi bersama (pendekatan NoSQL berbasis pasangan nilai utama), penyimpanan internal, pada kartu SD, dalam cache lokal, atau pada sistem jarak jauh. Ini, pada gilirannya, menentukan rangkaian aturan pemetaan yang dijelaskan sebelumnya yang harus digunakan. Persyaratan operasional lainnya adalah kemampuan untuk mengimpor dan mengekspor data menggunakan XML. Sekali lagi, ini dapat membatasi penyimpanan objek yang sedang dipertimbangkan.

Persyaratan kinerja utama yang mempengaruhi lapisan manajemen data adalah kecepatan dan kapasitas. Seperti dijelaskan sebelumnya, tergantung pada pola penggunaan yang diantisipasi—dan, setelah itu, aktual— dari objek yang disimpan, pendekatan pengindeksan dan caching yang berbeda mungkin diperlukan. Saat mempertimbangkan untuk mendistribusikan objek melalui jaringan, pertimbangan kecepatan dapat menyebabkan objek direplikasi pada node yang berbeda dalam jaringan. Dengan demikian, beberapa salinan dari objek yang sama dapat disimpan di lokasi yang berbeda di jaringan. Ini menimbulkan masalah anomali pembaruan yang dijelaskan sebelumnya dalam hubungannya dengan normalisasi. Bergantung pada aplikasi yang sedang dibangun, penyimpanan data NoSQL yang mendukung model pembaruan yang pada akhirnya konsisten mungkin sesuai. Juga, tergantung pada perkiraan ukuran dan pertumbuhan sistem, DBMS yang berbeda mungkin perlu dipertimbangkan. Persyaratan tambahan yang dapat memengaruhi desain lapisan manajemen data berkaitan dengan ketersediaan objek yang disimpan. Mungkin masuk akal untuk membatasi ketersediaan ke objek yang berbeda berdasarkan waktu. Misalnya, satu kelas pengguna mungkin diizinkan untuk mengakses satu set objek hanya dari jam 8 sampai 12 pagi dan kelompok pengguna kedua mungkin hanya bisa mengaksesnya dari jam 1 sampai jam 5 sore. Melalui DBMS, jenis pembatasan ini dapat diatur.

Persyaratan keamanan terutama berkaitan dengan kontrol akses, enkripsi, dan pencadangan. Melalui DBMS modern, berbagai jenis akses dapat diatur (misalnya, Baca, Perbarui, atau Hapus) yang memberikan akses hanya kepada pengguna (atau kelas pengguna) yang telah diotorisasi. Selanjutnya, kontrol akses dapat diatur untuk menjamin bahwa hanya pengguna dengan hak "administrator" yang diizinkan untuk mengubah skema penyimpanan objek atau kontrol akses. Persyaratan enkripsi pada lapisan ini berkaitan dengan apakah objek harus disimpan dalam format terenkripsi atau tidak. Meskipun objek terenkripsi lebih aman daripada objek tidak terenkripsi, proses enkripsi dan dekripsi objek akan memperlambat sistem. Tergantung pada arsitektur fisik yang digunakan, biaya enkripsi mungkin dapat diabaikan. Misalnya, jika kami berencana untuk mengenkripsi objek sebelum mengirimkannya melalui jaringan, mungkin tidak ada biaya tambahan untuk menyimpannya dalam format terenkripsi. Persyaratan cadangan berurusan dengan memastikan bahwa objek secara rutin disalin dan disimpan jika penyimpanan objek rusak atau tidak dapat digunakan. Membuat salinan cadangan secara berkala dan menyimpan pembaruan yang telah terjadi sejak salinan cadangan terakhir dibuat memastikan bahwa pembaruan tidak hilang dan

penyimpanan objek dapat disusun kembali dengan menjalankan salinan pembaruan terhadap salinan cadangan untuk membuat salinan baru saat ini.

Ada beberapa persyaratan politik dan budaya yang dapat mempengaruhi lapisan manajemen data. Ini termasuk masalah yang terkait dengan jumlah karakter yang diharapkan yang harus dialokasikan untuk bidang data, format bidang data, dan masalah yang terkait dengan keamanan. Misalnya, berapa banyak karakter yang harus dialokasikan untuk bidang nama belakang yang merupakan bagian dari objek Karyawan, format tanggal apa yang harus disimpan, atau di mana data akan ditempatkan secara fisik—bagian dunia yang berbeda memiliki undang-undang yang berbeda mengenai perlindungan data. Terakhir, mungkin ada bias TI perusahaan terhadap platform perangkat keras dan perangkat lunak yang berbeda. Jika demikian, ini dapat membatasi jenis penyimpanan objek yang tersedia.

9.8 VERIFIKASI DAN VALIDASI LAPISAN MANAJEMEN DATA

Seperti model pada lapisan domain masalah, spesifikasi untuk lapisan manajemen data perlu diverifikasi dan divalidasi. Sekarang, mungkin tampak agak berat untuk bersikeras lebih banyak memverifikasi dan memvalidasi. Namun, tergantung pada kegigihan objek yang dipilih, perubahan yang telah diterapkan pada desain sistem yang berkembang mungkin sangat substansial. Akibatnya, sangat penting untuk benar-benar menguji kesetiaan desain lagi sebelum sistem diimplementasikan. Tanpa menguji lapisan manajemen data secara menyeluruh, tidak ada jaminan bahwa sistem yang efisien dan efektif akan diterapkan. Memverifikasi dan memvalidasi desain lapisan manajemen data dibagi menjadi tiga kelompok dasar.

Pertama, kami merekomendasikan untuk memverifikasi dan memvalidasi setiap perubahan yang dibuat pada domain masalah dengan melakukan penelusuran model fungsional yang dimodifikasi (Bab 4), model struktural (Bab 5), dan model perilaku (Bab 6). Selanjutnya, semua model harus konsisten dan seimbang (Bab 7). Dan, jika ada kelas domain masalah yang dimodifikasi yang dikaitkan dengan skenario Use-Case, skenario itu harus diuji lagi melalui permainan peran.

Kedua, ketergantungan contoh persistensi objek pada domain masalah harus ditegakkan. Misalnya, semua invarian yang terkait dengan kelas domain masalah (Bab 8) perlu diverifikasi dan divalidasi. Misalnya, jika bidang data nama ditentukan dalam kelas domain masalah sebagai panjang tiga puluh lima karakter dan sebagai bidang wajib, maka batasan serupa harus diterapkan saat bidang disimpan.

Ketiga, desain kelas akses dan manipulasi data perlu diuji untuk memastikan bahwa mereka bergantung pada kelas domain masalah dan format persistensi objek, bukan sebaliknya. Sebagai contoh, pada Gambar 9-21, kita melihat bahwa kelas Patient-DAM bergantung pada kelas domain masalah Pasien dan tabel Pasien.

Setelah sistem diimplementasikan, pengujian lapisan manajemen data menjadi lebih penting. Salah satu masalah yang harus ditangani adalah pengujian persyaratan nonfungsional. Dalam hal ini, pengujian harus dirancang dan dilakukan untuk setiap persyaratan nonfungsional. Misalnya, untuk persyaratan kinerja, pengujian beban harus dilakukan untuk mengidentifikasi kemungkinan kemacetan kinerja dalam database. Kami akan kembali ke topik ini di Bab 12.

Pertanyaan

1. Jelaskan empat langkah dalam desain persistensi objek.
2. Bagaimana file dan database berbeda satu sama lain?
3. Apa perbedaan antara database pengguna akhir dan database perusahaan? Berikan contoh masing-masing.
4. Apa perbedaan antara file akses berurutan dan acak?
5. Sebutkan lima jenis file dan jelaskan tujuan utama dari setiap jenis.
6. Apa jenis database yang paling populer saat ini? Berikan tiga contoh produk yang didasarkan pada teknologi database ini.
7. Apa itu integritas referensial dan bagaimana penerapannya dalam RDBMS?
8. Sebutkan beberapa perbedaan antara ORDBMS dan RDBMS.
9. Apa keuntungan menggunakan ORDBMS dibandingkan RDBMS?
10. Sebutkan beberapa perbedaan antara ORDBMS dan OODBMS.
11. Apa keuntungan menggunakan ORDBMS dibandingkan OODBMS?
12. Apa keuntungan menggunakan OODBMS dibandingkan RDBMS?
13. Apa keuntungan menggunakan OODBMS dibandingkan ORDBMS?
14. Apa faktor-faktor dalam menentukan jenis format persistensi objek yang harus diadopsi untuk suatu sistem? Mengapa faktor-faktor ini begitu penting?
15. Mengapa Anda harus mempertimbangkan format penyimpanan yang sudah ada dalam suatu organisasi ketika memutuskan format penyimpanan untuk sistem baru?
16. Saat menerapkan persistensi objek dalam ORDBMS, jenis masalah apa yang harus Anda atasi?
17. Saat menerapkan persistensi objek dalam RDBMS, jenis masalah apa yang harus Anda atasi?
18. Sebutkan tiga cara nilai null dapat diinterpretasikan dalam database relasional. Mengapa ini bermasalah?
19. Apa dua dimensi untuk mengoptimalkan database relasional?
20. Apa tujuan dari normalisasi?
21. Bagaimana model memenuhi persyaratan bentuk normal ketiga?
22. Jelaskan tiga situasi yang dapat menjadi kandidat yang baik untuk denormalisasi.
23. Jelaskan beberapa teknik yang dapat meningkatkan kinerja database.
24. Apa perbedaan antara pengelompokan interfile dan intrafile? Mengapa mereka digunakan?
25. Apa itu indeks dan bagaimana cara meningkatkan kinerja sistem?
26. Jelaskan apa yang harus dipertimbangkan ketika memperkirakan ukuran database.
27. Mengapa penting untuk memahami ukuran awal dan proyeksi database selama desain?
28. Apa saja persyaratan nonfungsional yang dapat memengaruhi desain lapisan manajemen data?
29. Apa masalah utama dalam memutuskan antara menggunakan database yang dinormalisasi sempurna dan database yang didenormalisasi?
30. Apa tujuan utama dari kelas akses dan manipulasi data?
31. Mengapa kelas akses dan manipulasi data harus bergantung pada kelas domain masalah alih-alih sebaliknya?
32. Mengapa kelas kegigihan objek bergantung pada kelas domain masalah alih-alih sebaliknya?

Latihan

A. Menggunakan Web atau sumber daya lainnya, mengidentifikasi produk yang dapat diklasifikasikan sebagai database pengguna akhir dan produk yang dapat diklasifikasikan sebagai database perusahaan. Bagaimana produk dideskripsikan dan dipasarkan? Jenis aplikasi dan pengguna apa yang mereka dukung? Dalam situasi

seperti apa organisasi memilih untuk mengimplementasikan database pengguna akhir di atas database perusahaan?

B. Kunjungi situs web komersial (mis., Amazon.com). Jika file digunakan untuk menyimpan data yang mendukung aplikasi, jenis file apa yang diperlukan? Jenis akses apa yang diperlukan? Data apa yang akan mereka isi?
C. Menggunakan Web, tinjau salah satu produk berikut. Apa fitur dan fungsi utama dari perangkat lunak? Di perusahaan apa DBMS telah diimplementasikan, dan untuk tujuan apa? Berdasarkan informasi yang Anda temukan, apa tiga kekuatan dan kelemahan produk?
    a. DBMS relasional
    b. DBMS objek-relasional
    c. DBMS berorientasi objek
D. Anda telah diberikan file yang berisi bidang berikut yang berkaitan dengan informasi CD. Menggunakan langkah-langkah normalisasi, buat model yang merepresentasikan file ini dalam bentuk normal ketiga. Bidang tersebut meliputi:

Musical group name	CD title 2
Musicians in group	CD title 3
Date group was formed	CD 1 length
Group's agent	CD 2 length
CD title 1	CD 3 length

Asumsi:

- Musisi dalam grup berisi daftar anggota orang-orang dalam grup musik.
- Grup musik dapat memiliki lebih dari satu CD, sehingga nama grup dan judul CD diperlukan untuk mengidentifikasi CD tertentu secara unik.

E. Dealer Jim Smith menjual Ford, Honda, dan Toyota. Dealer menyimpan informasi tentang setiap produsen mobil yang berurusan dengannya sehingga dealer dapat menghubungi mereka dengan mudah. Diler juga menyimpan informasi model mobil yang dibawanya dari masing-masing pabrikan. Itu menyimpan informasi seperti daftar harga, harga yang dibayar dealer untuk mendapatkan model, dan nama model dan seri (misalnya, Honda Civic LX). Itu juga menyimpan informasi tentang semua penjualan yang telah dilakukan (misalnya, mencatat nama pembeli, mobil yang dibeli, dan jumlah yang dibayarkan untuk mobil). Untuk menghubungi pembeli di masa mendatang, informasi kontak juga disimpan (misalnya, alamat, nomor telepon). Buat diagram kelas untuk situasi ini. Terapkan aturan normalisasi ke diagram kelas untuk memeriksa diagram untuk efisiensi pemrosesan.
F. Jelaskan bagaimana Anda akan mendenormalisasi model yang Anda buat dalam latihan E. Gambarlah diagram kelas baru berdasarkan perubahan yang Anda sarankan. Bagaimana kinerja akan terpengaruh oleh saran Anda?
G. Periksa model yang Anda buat dalam latihan F. Kembangkan strategi pengelompokan dan pengindeksan untuk model ini. Jelaskan bagaimana strategi Anda akan meningkatkan kinerja database.
H. Hitung ukuran database yang Anda buat dalam latihan F. Berikan perkiraan ukuran untuk ukuran awal database serta untuk database dalam waktu satu tahun. Asumsikan bahwa dealer menjual sepuluh model mobil dari masing-masing pabrikan kepada sekitar 20.000 pelanggan per tahun. Sistem akan diatur awalnya dengan data senilai satu tahun.

I. Untuk masalah A Real Estate Inc. di Bab 4 (latihan I, J, dan K), Bab 5 (latihan P dan Q), Bab 6 (latihan D), Bab 7 (latihan A), dan Bab 8 (latihan A):
    a. Terapkan aturan normalisasi ke diagram kelas untuk memeriksa diagram untuk efisiensi pemrosesan.
    b. Kembangkan strategi pengelompokan dan pengindeksan untuk model ini. Jelaskan bagaimana strategi Anda akan meningkatkan kinerja database.
J. Untuk masalah A Video Store di Bab 4 (latihan L, M, dan N), Bab 5 (latihan R dan S), Bab 6 (latihan E), Bab 7 (latihan B), dan Bab 8 (latihan B):
    a. Terapkan aturan normalisasi ke diagram kelas untuk memeriksa diagram untuk efisiensi pemrosesan.
    b. Kembangkan strategi pengelompokan dan pengindeksan untuk model ini. Jelaskan bagaimana strategi Anda akan meningkatkan kinerja database.
K. Untuk masalah keanggotaan gym di Bab 4 (latihan O, P, dan Q), Bab 5 (latihan T dan U), Bab 6 (latihan F), Bab 7 (latihan C), dan Bab 8 (latihan C):
    a. Terapkan aturan normalisasi ke diagram kelas untuk memeriksa diagram untuk efisiensi pemrosesan.
    b. Kembangkan strategi pengelompokan dan pengindeksan untuk model ini. Jelaskan bagaimana strategi Anda akan meningkatkan kinerja database.
L. Untuk soal Piknik R Us di Bab 4 (latihan R, S, dan T), Bab 5 (latihan V dan W), Bab 6 (latihan G), Bab 7 (latihan D), dan Bab 8 (latihan D):
    a. Terapkan aturan normalisasi ke diagram kelas untuk memeriksa diagram untuk efisiensi pemrosesan.
    b. Kembangkan strategi pengelompokan dan pengindeksan untuk model ini. Jelaskan bagaimana strategi Anda akan meningkatkan kinerja database.
M. Untuk masalah Klub Bulanan di Bab 4 (latihan U, V, dan W), Bab 5 (latihan X dan Y), Bab 6 (latihan H), Bab 7 (latihan E), dan Bab 8 (latihan E):
    a. Terapkan aturan normalisasi ke diagram kelas untuk memeriksa diagram untuk efisiensi pemrosesan.
    b. Kembangkan strategi pengelompokan dan pengindeksan untuk model ini. Jelaskan bagaimana strategi Anda akan meningkatkan kinerja database.

Kasus Kecil

1. Tim pengembangan sistem di Perusahaan Wilcon sedang mengembangkan sistem entri pesanan pelanggan baru. Dalam proses merancang sistem baru, tim telah mengidentifikasi kelas berikut dan atributnya:

Inventaris
Memesan
Nomor Pesanan (PK)
Tanggal pemesanan
Nama Pelanggan
Alamat jalan
Kota
Negara
Kode Pos
Tipe pelanggan
Inisial
Nomor Distrik
Nomor Wilayah
1 sampai 22 kemunculan:

Nama barang Jumlah yang Dipesan Barang Satuan Kuantitas Dikirim Barang Keluar Jumlah Diterima

a. Nyatakan aturan yang diterapkan untuk menempatkan kelas dalam bentuk normal pertama. Berdasarkan class di atas, buatlah class diagram yang akan di 1NF.
b. Nyatakan aturan yang diterapkan untuk menempatkan suatu kelas ke dalam bentuk normal kedua. Merevisi diagram kelas untuk Perusahaan Wilcon menggunakan kelas dan atribut yang dijelaskan (jika perlu) untuk menempatkannya di 2NF.
c. Nyatakan aturan yang diterapkan untuk menempatkan suatu kelas ke dalam bentuk normal ketiga. Merevisi diagram kelas untuk menempatkannya di 3NF.
d. Saat merencanakan desain fisik database ini, dapatkah Anda mengidentifikasi kemungkinan situasi di mana tim proyek mungkin memilih untuk mendenormalisasi diagram kelas? Setelah melalui pekerjaan normalisasi, mengapa ini dipertimbangkan?

2. Dalam sistem baru yang sedang dikembangkan untuk Kendaraan Perjalanan Liburan, tujuh tabel akan diimplementasikan dalam database relasional yang baru. Tabel-tabel tersebut adalah: Kendaraan Baru, Kendaraan Tukar Tambah, Faktur Penjualan, Pelanggan, Tenaga Penjual, Opsi Terpasang, dan Opsi. Ukuran catatan rata-rata yang diharapkan untuk tabel ini dan jumlah catatan awal per tabel diberikan di sini.

Table Name	Average Record Size	Initial Table Size (records)
New Vehicle	65 characters	10,000
Trade-in Vehicle	48 characters	7,500
Sales Invoice	76 characters	16,000
Customer	61 characters	13,000
Salesperson	34 characters	100
Installed Option	16 characters	25,000
Option	28 characters	500

Lakukan analisis volumetrik untuk sistem Kendaraan Perjalanan Liburan. Asumsikan bahwa DBMS yang akan digunakan untuk mengimplementasikan sistem membutuhkan 35 persen overhead untuk diperhitungkan dalam perkiraan. Juga, asumsikan tingkat pertumbuhan perusahaan sebesar 10 persen per tahun. Tim pengembangan sistem ingin memastikan bahwa perangkat keras yang memadai diperoleh untuk tiga tahun ke depan.

3. Lihat minicase Professional and Scientific Staff Management (PSSM) di Bab 4, 6, 7, dan 8.
   a. Terapkan aturan normalisasi ke diagram kelas untuk memeriksa diagram untuk efisiensi pemrosesan.
   b. Kembangkan strategi pengelompokan dan pengindeksan untuk model ini. Jelaskan bagaimana strategi Anda akan meningkatkan kinerja database.

BAB 10

DESAIN LAPISAN INTERAKSI MANUSIA-KOMPUTER

Antarmuka pengguna adalah bagian dari sistem tempat pengguna berinteraksi. Dari sudut pandang pengguna, antarmuka pengguna adalah sistem. Ini mencakup tampilan layar yang menyediakan navigasi melalui sistem, layar dan formulir yang menangkap data, dan laporan yang dihasilkan sistem (baik di atas kertas, di layar, atau melalui media lain). Bab ini memperkenalkan prinsip-prinsip dasar dan proses desain antarmuka dan membahas bagaimana merancang struktur dan standar antarmuka, desain navigasi, desain input, dan desain output. Bab ini memperkenalkan isu-isu yang berkaitan dengan merancang antarmuka pengguna untuk lingkungan komputasi mobile dan media sosial. Ini juga memperkenalkan isu-isu yang perlu dipertimbangkan ketika merancang antarmuka pengguna untuk audiens global. Akhirnya, bab ini menjelaskan pengaruh persyaratan nonfungsional pada perancangan lapisan interaksi manusia-komputer.

10.1 Tujuan

- Memahami beberapa prinsip dasar desain antarmuka pengguna/User Interface (UI).
- Memahami proses desain antarmuka pengguna.
- Memahami bagaimana merancang struktur antarmuka pengguna.
- Memahami bagaimana merancang standar antarmuka pengguna.
- Memahami prinsip dan teknik yang umum digunakan untuk desain navigasi.
- Memahami prinsip dan teknik yang umum digunakan untuk desain input.
- Memahami prinsip dan teknik yang umum digunakan untuk desain Output.
- Mampu merancang antarmuka pengguna.
- Memahami efek persyaratan nonfungsional pada lapisan interaksi manusia-komputer.

10.2 Pendahuluan

Desain antarmuka adalah proses mendefinisikan bagaimana sistem akan berinteraksi dengan entitas eksternal (misalnya, pelanggan, pemasok, sistem lain). Dalam bab ini, kami fokus pada desain antarmuka pengguna, tetapi juga penting untuk diingat bahwa terkadang ada antarmuka sistem, yang bertukar informasi dengan sistem lain. Antarmuka sistem biasanya dirancang sebagai bagian dari upaya integrasi sistem. Mereka didefinisikan secara umum sebagai bagian dari arsitektur fisik dan lapisan manajemen data. Lapisan interaksi manusia-komputer mendefinisikan cara di mana pengguna berinteraksi dengan sistem dan sifat input dan output yang diterima dan dihasilkan sistem.

Hingga saat ini, seluruh proses pengembangan telah difokuskan untuk mendapatkan lapisan domain masalah dan penyimpanannya pada lapisan manajemen data dengan benar. Namun, dari sudut pandang pengguna, antarmuka pengguna pada lapisan interaksi manusia-komputer adalah sistem. Pengguna tidak terlalu peduli tentang bagaimana objek domain masalah disimpan. Namun, mereka peduli tentang bagaimana mereka dapat menggunakan sistem untuk mendukung aktivitas mereka. Berdasarkan pendekatan desain berbasis berlapis kami, antarmuka pengguna dari lapisan interaksi manusia-komputer tidak tergantung pada lapisan manajemen data. Tapi itu tergantung pada domain masalah dan lapisan arsitektur fisik. Bergantung pada jenis perangkat yang digunakan lapisan interaksi manusia-komputer akan menetapkan peluang dan kendala untuk fitur antarmuka pengguna apa yang dapat disertakan. Misalnya, menerapkan lapisan interaksi komputer manusia pada smartphone dan komputer desktop akan menyebabkan dua antarmuka pengguna yang berbeda dirancang.

Meskipun ada antarmuka pengguna baris perintah (misalnya, Terminal pada Mac OSX), kami hanya berfokus pada antarmuka pengguna grafis/*graphical user interfaces* (GUI) yang menggunakan jendela, menu, ikon, dll. Saat ini, antarmuka berbasis GUI adalah yang paling umum jenis antarmuka yang kami gunakan. Terlepas dari perangkat keras yang digunakan, antarmuka pengguna berbasis GUI terdiri dari tiga bagian mendasar. Yang pertama adalah mekanisme navigasi, cara di mana pengguna memberikan instruksi ke sistem dan memberitahu apa yang harus dilakukan (misalnya, tombol, menu). Yang kedua adalah mekanisme input, cara sistem menangkap informasi (misalnya, formulir untuk menambah pelanggan baru). Yang ketiga adalah mekanisme Output, cara sistem memberikan informasi kepada pengguna atau sistem lain (misalnya, laporan, halaman Web). Masing-masing secara konseptual berbeda, tetapi mereka saling terkait erat. Semua tampilan berbasis GUI berisi mekanisme navigasi, dan sebagian besar berisi mekanisme input dan output. Oleh karena itu, desain navigasi, desain input, dan desain output digabungkan secara erat dan harus dilakukan secara inkremental dan iteratif.

Dalam bab ini, meskipun kami berfokus terutama pada perancangan antarmuka pengguna yang berjalan di lingkungan jenis laptop atau desktop, kami juga memberikan panduan umum untuk komputasi seluler. Kami juga mengatasi beberapa masalah unik yang Anda hadapi saat menerapkan antarmuka pengguna di aplikasi sosial, seperti Facebook™ dan Twitter™; dalam antarmuka teknologi canggih, seperti augmented 3D dan aplikasi realitas virtual; dan terakhir, isu-isu yang terkait dengan global dengan antarmuka pengguna.

10.3 PRINSIP UNTUK DESAIN ANTARMUKA PENGGUNA (USER INTERFACE/UI)

Dalam banyak hal, desain antarmuka pengguna adalah seni. Tujuannya adalah untuk membuat antarmuka menyenangkan mata dan mudah digunakan sambil meminimalkan upaya yang dibutuhkan pengguna untuk menyelesaikan pekerjaan mereka. Sistem tidak pernah menjadi tujuan itu sendiri; itu hanyalah sarana untuk mencapai bisnis organisasi.

Kami telah menemukan bahwa masalah terbesar yang dihadapi desainer berpengalaman adalah menggunakan ruang secara efektif. Sederhananya, seringkali ada lebih banyak informasi yang perlu disajikan di layar atau laporan atau formulir daripada yang sesuai dengan nyaman. Analis harus menyeimbangkan kebutuhan akan kesederhanaan dan penampilan yang menyenangkan dengan kebutuhan untuk menyajikan informasi di beberapa halaman atau layar, yang mengurangi kesederhanaan. Pada bagian ini, kita membahas beberapa prinsip dasar desain antarmuka, yang umum untuk desain navigasi, desain input, dan desain output3 (lihat Gambar 10-1).

Prinsip	Deskripsi
Layout	Antarmuka harus berupa serangkaian area pada layar yang digunakan secara konsisten untuk tujuan yang berbeda—misalnya, area atas untuk perintah dan navigasi, area tengah untuk informasi yang akan dimasukkan atau dikeluarkan, dan area bawah untuk informasi status.
Content Awareness	Pengguna harus selalu menyadari di mana mereka berada dalam sistem dan informasi apa yang ditampilkan.
Aesthetics	Antarmuka harus fungsional dan mengundang pengguna melalui penggunaan spasi, warna, dan font yang cermat. Seringkali ada trade-off antara memasukkan ruang putih yang cukup untuk membuat antarmuka terlihat menyenangkan tanpa kehilangan begitu banyak ruang sehingga informasi penting tidak muat di layar.

User Experience	Meskipun kemudahan penggunaan dan kemudahan belajar sering mengarah pada keputusan desain yang serupa, terkadang ada trade-off antara keduanya. Pengguna perangkat lunak pemula atau jarang lebih menyukai kemudahan belajar, sedangkan pengguna yang sering lebih menyukai kemudahan penggunaan.
Consistency	Konsistensi dalam desain antarmuka memungkinkan pengguna untuk memprediksi apa yang akan terjadi sebelum mereka melakukan suatu fungsi. Ini adalah salah satu elemen terpenting dalam kemudahan belajar, kemudahan penggunaan, dan estetika.
Minimal User Effort	Antarmuka harus mudah digunakan. Kebanyakan desainer berencana untuk memiliki tidak lebih dari tiga klik mouse dari menu awal sampai pengguna melakukan pekerjaan

Gambar 10-1 Prinsip Desain Antarmuka Pengguna

Layout

Elemen pertama dari desain adalah tata letak dasar layar, formulir, atau laporan. Sebagian besar perangkat lunak yang dirancang untuk komputer pribadi mengikuti pendekatan standar Windows atau Macintosh untuk desain layar. Layar dibagi menjadi tiga kotak. Kotak atas adalah area navigasi, di mana pengguna mengeluarkan perintah untuk menavigasi melalui sistem. Kotak bawah adalah area status, yang menampilkan informasi tentang apa yang dilakukan pengguna. Kotak menganggur—dan terbesar—digunakan untuk menampilkan laporan dan menyajikan formulir untuk entri data.

Penggunaan beberapa area tata letak ini juga berlaku untuk input dan output. Area data pada laporan dan formulir sering dibagi lagi menjadi subarea, yang masing-masing digunakan untuk jenis informasi yang berbeda. Area-area ini hampir selalu berbentuk persegi panjang, meskipun terkadang batasan ruang membutuhkan bentuk yang aneh. Meskipun demikian, margin di tepi layar harus konsisten. Setiap area dalam laporan atau formulir dirancang untuk menyimpan informasi yang berbeda. Misalnya, pada formulir pemesanan (atau laporan pesanan), satu bagian dapat digunakan untuk informasi pelanggan (misalnya, nama, alamat), satu bagian untuk informasi tentang pesanan secara umum (misalnya, tanggal, informasi pembayaran), dan satu bagian lagi. untuk rincian pesanan (misalnya, berapa banyak unit barang apa dengan harga berapa masing-masing). Setiap area bersifat mandiri sehingga informasi di satu area tidak bertemu dengan area lain.

Area dan informasi di dalam area harus memiliki aliran intuitif alami untuk meminimalkan pergerakan pengguna dari satu area ke area berikutnya. Orang-orang di beberapa negara cenderung membaca kiri-ke-kanan, atas-ke-bawah, sehingga informasi terkait harus ditempatkan sehingga digunakan dalam urutan ini (misalnya, baris alamat, diikuti menurut kota, negara atau provinsi, lalu kode pos). Kadang-kadang urutannya berurutan, atau dari umum ke khusus, atau dari yang paling sering ke yang paling jarang digunakan. Bagaimanapun, sebelum area ditempatkan pada formulir atau laporan, analis harus memiliki pemahaman yang jelas tentang pengaturan apa yang paling masuk akal untuk bagaimana formulir atau laporan akan digunakan. Aliran antar bagian juga harus konsisten, baik horizontal maupun vertikal. Idealnya, area akan tetap konsisten dalam ukuran, bentuk, dan penempatan formulir yang digunakan untuk memasukkan informasi (baik kertas atau layar) dan laporan yang digunakan untuk menyajikannya.

Kesadaran Konten (Content Awareness)

Kesadaran konten mengacu pada kemampuan antarmuka untuk membuat pengguna mengetahui informasi yang dikandungnya dengan sedikit usaha dari pihak pengguna. Semua bagian antarmuka, baik navigasi, input, atau output, harus memberikan kesadaran konten sebanyak mungkin, tetapi ini sangat penting untuk formulir atau laporan yang digunakan dengan cepat atau tidak teratur. Kesadaran konten berlaku untuk antarmuka secara umum. Semua antarmuka harus memiliki judul (pada bingkai layar, misalnya). Menu harus menunjukkan di mana pengguna berada dan, jika mungkin, dari mana pengguna datang untuk sampai ke sana.

Kesadaran konten juga berlaku untuk area dalam formulir dan laporan. Semua area harus jelas dan terdefinisi dengan baik sehingga sulit bagi pengguna untuk menjadi bingung tentang informasi di area mana pun. Kemudian pengguna dapat dengan cepat menemukan bagian formulir atau laporan yang kemungkinan berisi informasi yang mereka butuhkan. Terkadang area ditandai dengan garis, warna, atau judul; dalam kasus lain, area tersebut hanya tersirat.

Kesadaran konten juga berlaku untuk bidang dalam setiap area. Fields adalah elemen individual dari data yang merupakan input atau output. Label bidang yang mengidentifikasi bidang pada antarmuka harus pendek dan spesifik—tujuan yang sering bertentangan. Seharusnya tidak ada ketidakpastian tentang format informasi dalam bidang, baik untuk entri atau tampilan. Misalnya, tanggal 10/5/15 berbeda tergantung apakah Anda berada di Indonesia Serikat (5 Oktober 2015) atau di Singapura (10 Mei 2015). Bidang apa pun yang ada kemungkinan ketidakpastian atau multitafsir harus memberikan penjelasan yang eksplisit.

Kesadaran konten juga berlaku untuk informasi yang berisi formulir atau laporan. Secara umum, semua formulir dan laporan harus berisi tanggal persiapan (yaitu, tanggal dicetak atau tanggal selesai) sehingga usia informasi jelas. Demikian juga, semua formulir dan perangkat lunak yang dicetak harus menyediakan nomor versi sehingga pengguna, analis, dan pemrogram dapat mengidentifikasi materi yang sudah ketinggalan zaman.

Estetika

Estetika mengacu pada desain antarmuka yang menyenangkan mata. Antarmuka tidak harus berupa karya seni, tetapi harus fungsional dan menarik untuk digunakan. Dalam kebanyakan kasus, less is more, artinya desain sederhana dan minimalis adalah yang terbaik.

Ruang biasanya sangat mahal pada formulir dan laporan, dan sering kali ada godaan untuk memasukkan informasi sebanyak mungkin ke halaman atau layar. Sayangnya, ini dapat membuat formulir atau laporan menjadi tidak menyenangkan sehingga pengguna tidak ingin menggunakannya. Secara umum, semua formulir dan laporan membutuhkan ruang kosong minimum yang sengaja dikosongkan.

Secara umum, pengguna antarmuka pemula atau jarang, baik di layar atau di atas kertas, lebih memilih antarmuka dengan kepadatan rendah, sering kali dengan kepadatan kurang dari 50 persen (yaitu, kurang dari 50 persen antarmuka ditempati oleh informasi). Pengguna yang lebih berpengalaman lebih menyukai kepadatan yang lebih tinggi, terkadang mendekati 90 persen terisi, karena mereka tahu di mana informasi berada dan kepadatan tinggi mengurangi jumlah pergerakan fisik melalui antarmuka.

Desain teks sama pentingnya. Sebagai aturan umum, semua teks harus dalam font yang sama dan ukuran yang hampir sama. Font tidak boleh lebih kecil dari 8 poin, tetapi 10 poin sering lebih disukai, terutama jika antarmuka akan digunakan oleh orang yang lebih tua.

Perubahan font dan ukuran digunakan untuk menunjukkan perubahan jenis informasi yang disajikan (misalnya, judul, indikator status). Secara umum, huruf miring dan garis bawah harus dihindari karena membuat teks lebih sulit dibaca.

Font serif (yaitu, yang memiliki huruf dengan serif, atau ekor, seperti Times Roman) adalah yang paling mudah dibaca untuk laporan tercetak, terutama untuk huruf kecil. Font sans serif (yaitu, font tanpa serif, seperti Helvetica atau Arial) adalah yang paling mudah dibaca untuk layar komputer dan sering digunakan untuk judul dalam laporan tercetak. Jangan pernah menggunakan huruf kapital semua, kecuali mungkin untuk judul.

Warna dan pola harus digunakan dengan hati-hati dan hemat dan hanya jika sesuai dengan tujuan. (Sekitar 10 persen pria buta warna, sehingga penggunaan warna yang tidak tepat dapat mengganggu kemampuan mereka untuk membaca teks berwarna.) Sebuah perjalanan singkat di Web akan menunjukkan masalah yang disebabkan oleh penggunaan warna dan pola yang sembarangan. Ingat, tujuannya adalah keterbacaan yang menyenangkan, bukan seni; warna dan pola harus digunakan untuk memperkuat pesan, bukan membanjirinya. Warna paling baik digunakan untuk memisahkan dan mengkategorikan item, seperti menunjukkan perbedaan antara judul dan teks biasa, atau untuk menyorot informasi penting. Oleh karena itu, warna dengan kontras tinggi harus digunakan (misalnya, hitam dan putih). Secara umum, teks hitam dengan latar belakang putih adalah yang paling mudah dibaca, dan biru di atas merah adalah yang paling tidak mudah dibaca. Juga, dalam hal penggunaan warna yang tepat, masalah budaya ikut bermain. Kita bahas ini nanti di bab ini.

Pengalaman Pengguna (Eser Experience/UX)

Pengalaman pengguna pada dasarnya dapat dipecah menjadi dua tingkat: mereka yang memiliki pengalaman dan yang tidak. Antarmuka harus dirancang untuk kedua jenis pengguna. Pengguna pemula biasanya paling memperhatikan kemudahan belajar—seberapa cepat mereka dapat mempelajari sistem baru. Pengguna ahli biasanya paling memperhatikan kemudahan penggunaan—seberapa cepat mereka dapat menggunakan sistem setelah mereka mempelajari cara menggunakannya. Seringkali keduanya saling melengkapi dan mengarah pada keputusan desain yang serupa, tetapi terkadang ada trade-off. Pemula, misalnya, sering kali lebih memilih menu yang menampilkan semua fungsi sistem yang tersedia, karena menu ini meningkatkan kemudahan belajar. Para ahli, di sisi lain, terkadang lebih memilih menu yang lebih sedikit yang diatur di sekitar fungsi yang paling umum digunakan.

Sistem yang pada akhirnya akan digunakan oleh banyak orang setiap hari lebih cenderung memiliki mayoritas pengguna ahli (mis., Sistem entri pesanan). Meskipun antarmuka harus mencoba untuk menyeimbangkan kemudahan penggunaan dan kemudahan belajar, jenis sistem ini harus lebih menekankan pada kemudahan penggunaan daripada kemudahan belajar. Pengguna harus dapat mengakses fungsi yang umum digunakan dengan cepat, dengan sedikit penekanan tombol atau sejumlah kecil pilihan menu.

Di banyak sistem lain (mis., Sistem pendukung keputusan), kebanyakan orang tetap menjadi pengguna sesekali selama masa pakai sistem. Dalam hal ini, penekanan yang lebih besar dapat ditempatkan pada kemudahan belajar daripada kemudahan penggunaan.

Kemudahan penggunaan dan kemudahan belajar sering kali berjalan beriringan—tetapi terkadang tidak. Penelitian menunjukkan bahwa pengguna ahli dan pemula memiliki persyaratan dan pola perilaku yang berbeda dalam beberapa kasus. Misalnya, pemula hampir tidak pernah melihat bagian bawah layar yang menyajikan informasi status, sedangkan para ahli merujuk ke bilah status saat mereka membutuhkan informasi. Sebagian besar sistem harus dirancang untuk mendukung pengguna yang sering, kecuali untuk sistem yang dirancang untuk jarang digunakan atau ketika diharapkan banyak pengguna baru atau pengguna sesekali.

Demikian juga, sistem yang berisi fungsionalitas yang hanya digunakan sesekali harus berisi antarmuka yang sangat intuitif atau antarmuka yang berisi panduan yang jelas dan eksplisit mengenai penggunaannya. Keseimbangan akses cepat ke fungsi dan panduan yang umum digunakan dan terkenal melalui fungsi baru dan yang kurang terkenal merupakan tantangan bagi perancang antarmuka, dan keseimbangan ini sering kali membutuhkan solusi yang elegan.

Konsistensi

Konsistensi dalam desain mungkin merupakan satu-satunya faktor terpenting dalam membuat sistem mudah digunakan karena memungkinkan pengguna untuk memprediksi apa yang akan terjadi. Ketika antarmuka konsisten, pengguna dapat berinteraksi dengan satu bagian dari sistem dan kemudian mengetahui cara berinteraksi dengan bagian lainnya, selain elemen unik untuk bagian tersebut. Konsistensi biasanya mengacu pada antarmuka dalam satu sistem komputer, sehingga semua bagian dari sistem yang sama bekerja dengan cara yang sama. Saya setuju, sistem juga harus konsisten dengan sistem komputer lain dalam organisasi dan dengan perangkat lunak komersial yang digunakan. Banyak alat pengembangan perangkat lunak mendukung antarmuka sistem yang konsisten dengan menyediakan objek antarmuka standar (misalnya, kotak daftar, menu tarik-turun, dan tombol radio).

Konsistensi terjadi pada banyak tingkat yang berbeda. Konsistensi dalam kontrol navigasi menyampaikan bagaimana tindakan dalam sistem harus dilakukan. Misalnya, menggunakan ikon atau perintah yang sama untuk mengubah item dengan jelas mengomunikasikan bagaimana perubahan dilakukan di seluruh sistem. Konsistensi dalam terminologi juga penting. Ini mengacu pada penggunaan kata yang sama untuk elemen pada formulir dan laporan (misalnya, bukan pelanggan di satu tempat dan klien di tempat lain). Kami juga percaya bahwa konsistensi dalam desain laporan dan formulir adalah penting, meskipun sebuah penelitian menunjukkan bahwa terlalu konsisten dapat menyebabkan masalah. Ketika laporan dan formulir sangat mirip kecuali untuk perubahan judul yang sangat kecil, pengguna terkadang salah menggunakan formulir yang salah dan memasukkan data yang salah atau salah menafsirkan informasinya. Implikasi untuk desain adalah membuat laporan dan formulir serupa tetapi memberi mereka beberapa elemen khusus (misalnya, warna, ukuran judul) yang memungkinkan pengguna untuk segera mendeteksi perbedaan.

Meminimalkan Upaya Pengguna

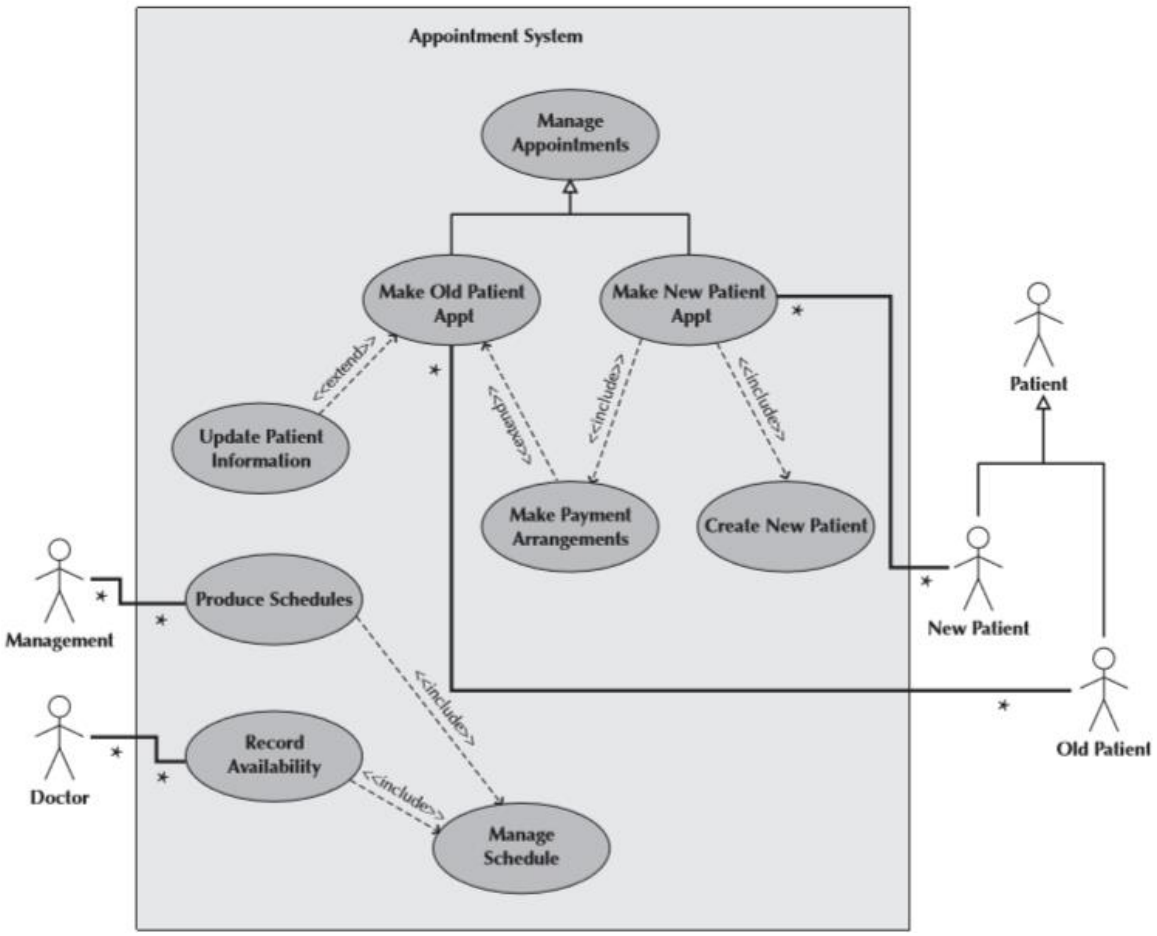
Antarmuka harus dirancang untuk meminimalkan jumlah upaya yang diperlukan untuk menyelesaikan tugas. Ini berarti menggunakan klik mouse atau penekanan tombol sesedikit mungkin untuk berpindah dari satu bagian sistem ke bagian lain. Sebagian besar desainer antarmuka mengikuti aturan tiga klik: Pengguna harus dapat beralih dari menu awal atau utama sistem ke informasi atau tindakan yang mereka inginkan dalam tidak lebih dari tiga klik mouse atau tiga penekanan tombol. Namun, sehubungan dengan hal ini, Anda perlu mengetahui prinsip-prinsip Krug (dibahas nanti).

10.4 PROSES DESAIN ANTARMUKA PENGGUNA/ USER INTERFACE (UI)

Desain antarmuka pengguna adalah proses yang didorong oleh Use-Case, inkremental, dan berulang. Analis sering bergerak bolak-balik antara bagian yang berbeda (navigasi, input, dan output) dari antarmuka pengguna, daripada melanjutkan secara berurutan dari satu bagian ke bagian lain. Mengingat bahwa proses desain didorong oleh Use-Case, analis memulai proses desain antarmuka pengguna dengan memeriksa Use-Case (lihat Bab 4) dan diagram urutan yang terkait (lihat Bab 6) yang dikembangkan dalam analisis. Analis kemudian biasanya duduk bersama pengguna untuk mengembangkan skenario penggunaan yang menggambarkan pola tindakan yang umum digunakan yang akan dilakukan pengguna sehingga antarmuka memungkinkan pengguna untuk melakukan skenario ini dengan cepat

dan lancar. Dalam beberapa kasus, persyaratan tambahan ditemukan. Tergantung pada pentingnya persyaratan yang baru ditemukan, ini dapat menyebabkan lapisan domain masalah dimodifikasi, yang pada gilirannya dapat menyebabkan lapisan manajemen data dimodifikasi. Namun, seringkali, persyaratan baru ini dapat ditunda hingga iterasi sistem berikutnya. Dengan pendekatan tangkas, desain antarmuka pengguna dan pemodelan persyaratan sangat terkait sehingga persyaratan baru ditemukan secara teratur. Akibatnya, tergantung pada stabilitas pemodelan domain masalah, desain antarmuka pengguna dapat terjadi bersamaan dengan pemodelan fungsional. Meskipun pemodelan fungsional dan perilaku dikaitkan dengan alur kerja analisis dan desain antarmuka pengguna dikaitkan dengan alur kerja desain, tingkat aktivitas yang terkait dengan dua alur kerja tumpang tindih (lihat Gambar 1-15 dan 1-16). Dengan demikian, melakukan desain antarmuka pengguna di sepanjang sisi pemodelan fungsional dan perilaku kompatibel dengan Proses Terpadu dan Proses Terpadu yang Ditingkatkan.

Setelah serangkaian skenario penggunaan dasar telah dikembangkan, antarmuka pengguna yang sebenarnya dirancang. Seperti yang kami nyatakan sebelumnya, semua antarmuka pengguna berbasis GUI terdiri dari tiga bagian: navigasi, input, dan output. Sampai taraf tertentu, ketiga bagian tersebut cenderung dirancang bersama. Akibatnya, proses desain antarmuka pengguna cenderung mengikuti gaya pengembangan prototipe (lihat Bab 1) di mana analis dan pengguna akan secara bertahap membangun desain dengan mengulangi ketiga bagian antarmuka pengguna menggunakan alat desain yang berbeda. Misalnya, ketika merancang struktur navigasi, diagram navigasi windows/*windows navigation diagram* (WND) sangat berguna; saat mendesain tata letak antarmuka pengguna, diagram tata letak windows sangat berguna; dan ketika mencoba untuk menggabungkan desain navigasi, input, dan output, storyboard, dan prototipe antarmuka pengguna sangat berguna. Ide lain yang berguna ketika mengembangkan antarmuka pengguna adalah memiliki seperangkat standar antarmuka yang diterima yang dapat digunakan di beberapa aplikasi. Misalnya, satu set standar menu, ikon, dan templat antarmuka pengguna menyederhanakan seluruh desain lapisan interaksi komputer manusia. Setelah desain dasar telah selesai untuk Use-Case tertentu, maka Use-Case penting yang dikembangkan dalam pemodelan fungsional harus diubah menjadi Use-Case nyata yang, bersama dengan alat lain yang digunakan untuk mendesain antarmuka, dapat digunakan sebagai dasar untuk dokumentasi, pelatihan, dan pengujian. Akhirnya, antarmuka individu dikenakan evaluasi antarmuka untuk menentukan apakah mereka memuaskan dan bagaimana mereka dapat ditingkatkan.

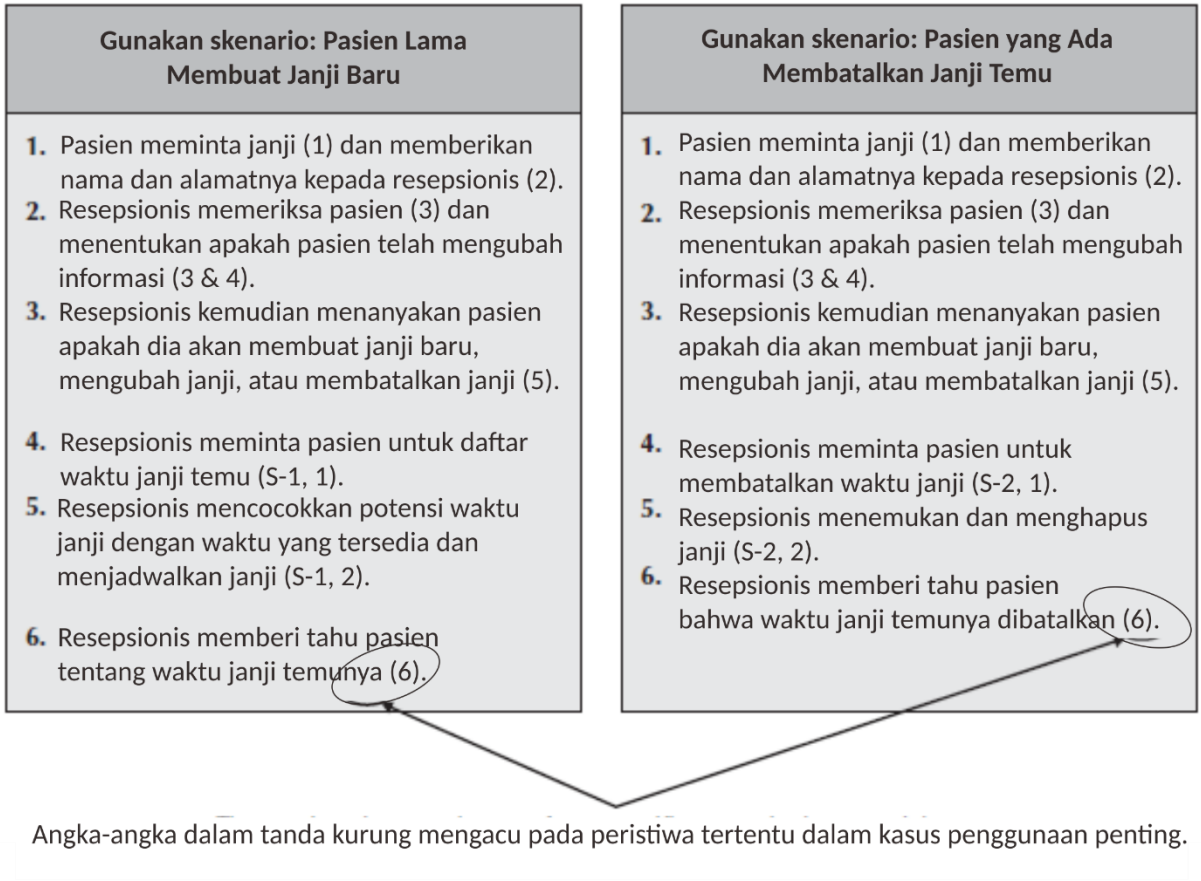


Gambar 10-2 Diagram Use-Case Sistem Penunjukan (lihat Gambar 4-21 dan 7-11)

Evaluasi antarmuka hampir selalu mengidentifikasi perbaikan, sehingga proses desain antarmuka diulang dalam proses siklus sampai tidak ada perbaikan baru yang diidentifikasi. Dalam praktiknya, sebagian besar analis berinteraksi erat dengan pengguna selama proses desain antarmuka sehingga pengguna memiliki banyak kesempatan untuk melihat antarmuka saat berkembang, daripada menunggu satu evaluasi antarmuka secara keseluruhan di akhir proses desain antarmuka. Lebih baik bagi semua pihak (baik analis dan pengguna) jika perubahan diidentifikasi lebih cepat daripada nanti. Misalnya, jika struktur antarmuka atau standar memerlukan perbaikan, lebih baik untuk mengidentifikasi perubahan sebelum sebagian besar layar yang menggunakan standar telah dirancang.

Gunakan Pengembangan Skenario

Skenario penggunaan adalah garis besar langkah-langkah yang dilakukan pengguna untuk menyelesaikan beberapa bagian dari pekerjaan mereka. Skenario penggunaan adalah satu jalur melalui Use-Case yang penting. Misalnya, Gambar 10-2 menunjukkan diagram Use-Case untuk Sistem Pengangkatan. Gambar ini menunjukkan bahwa use case Buat Pasien Baru berbeda dari use case Lakukan Pengaturan Pembayaran. Kami memodelkan dua Use-Case ini secara terpisah karena keduanya mewakili proses terpisah yang digunakan oleh Use-Case Aplikasi Buat Pasien Baru.



Gambar 10-3 Gunakan Skenario

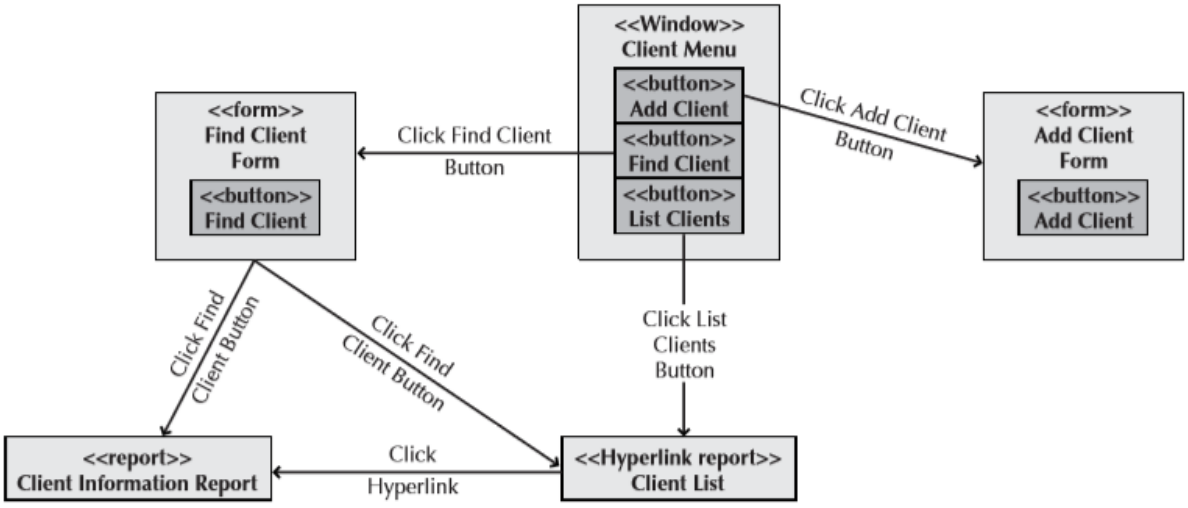
Diagram use-case dirancang untuk memodelkan semua kemungkinan penggunaan sistem — fungsionalitas lengkapnya atau semua jalur yang mungkin melalui use case pada tingkat abstraksi yang cukup tinggi. Dalam satu skenario penggunaan, pasien membuat permintaan dengan resepsionis mengenai janji dengan dokter. Resepsionis memeriksa pasien dan memeriksa apakah pasien memiliki tagihan yang harus dibayar. Resepsionis kemudian bertanya kepada pasien apakah dia ingin membuat janji baru, membatalkan janji yang sudah ada, atau mengubah janji yang sudah ada. Jika pasien ingin membuat janji baru, resepsionis meminta pasien untuk beberapa waktu janji temu yang disarankan, yang dicocokkan oleh resepsionis dengan waktu potensial yang tersedia. Resepsionis akhirnya membuat janji baru (lihat Gambar 6-1 dan 6-10).

Dalam skenario penggunaan lain, pasien hanya ingin membatalkan janji. Dalam hal ini, resepsionis mencari pasien dan memeriksa apakah pasien memiliki tagihan yang harus dibayar. Resepsionis kemudian meminta pasien untuk membatalkan waktu janji temu. Akhirnya, resepsionis menghapus janji temu.

Skenario penggunaan disajikan dalam deskripsi naratif sederhana yang terkait dengan Use-Case penting yang dikembangkan selama analisis (lihat Bab 4). Gambar 10-3 menunjukkan dua skenario penggunaan yang baru saja dijelaskan. Poin utama dengan use case untuk desain antarmuka bukanlah untuk mendokumentasikan semua skenario penggunaan yang mungkin dalam use case. Tujuannya adalah untuk mendokumentasikan dua atau tiga skenario penggunaan yang paling umum sehingga antarmuka dapat dirancang untuk memungkinkan penggunaan yang paling umum dilakukan dengan sederhana dan mudah.

Desain Struktur Navigasi

Struktur navigasi mendefinisikan komponen dasar antarmuka dan bagaimana mereka bekerja sama untuk menyediakan fungsionalitas bagi pengguna. Diagram navigasi windows (WND)6 digunakan untuk menunjukkan bagaimana semua layar, formulir, dan laporan yang digunakan oleh sistem terkait dan bagaimana pengguna berpindah dari satu ke yang lain. Kebanyakan sistem memiliki beberapa WND, satu untuk setiap bagian utama dari sistem.



Gambar 10-4 Contoh WND

WND sangat mirip dengan mesin status perilaku (lihat Bab 6), karena keduanya memodelkan perubahan status. Mesin status perilaku biasanya memodelkan perubahan status suatu objek, sedangkan WND memodelkan perubahan status antarmuka pengguna. Dalam WND, setiap status antarmuka pengguna direpresentasikan sebagai sebuah kotak. Kotak biasanya sesuai dengan komponen antarmuka pengguna, seperti jendela, formulir, tombol, atau laporan. Sebagai contoh, pada Gambar 10-4, ada lima status terpisah: Menu Klien, Formulir Temukan Klien, Formulir Tambah Klien, Daftar Klien, dan Laporan Informasi Klien.

Transisi dimodelkan sebagai panah berkepala tunggal atau berkepala dua. Panah berkepala tunggal menunjukkan bahwa pengembalian ke status panggilan tidak diperlukan, sedangkan panah berkepala dua menunjukkan pengembalian yang diperlukan. Misalnya pada Gambar 10-4, transisi dari status Menu Klien ke status Temukan Formulir Klien tidak memerlukan pengembalian. Panah diberi label dengan tindakan yang menyebabkan antarmuka pengguna berpindah dari satu keadaan ke keadaan lain. Misalnya, pada Gambar 10-4, untuk berpindah dari status Menu Klien ke status Temukan Formulir Klien, pengguna harus mengklik Tombol Temukan Klien pada Menu Klien.

Item terakhir yang akan dijelaskan dalam WND adalah stereotip. Stereotip dimodelkan sebagai item teks yang diapit oleh guillemet atau kurung sudut (<< >>). Stereotip mewakili jenis komponen antarmuka pengguna dari sebuah kotak pada diagram. Misalnya, Menu Klien adalah jendela, sedangkan Formulir Temukan Klien adalah formulir.

Struktur navigasi dasar dari sebuah antarmuka mengikuti struktur dasar dari proses bisnis itu sendiri, seperti yang didefinisikan dalam Use-Case dan model perilaku. Analis mulai dengan Use-Case penting dan mengembangkan aliran dasar kontrol sistem saat bergerak dari objek ke objek. Analis kemudian memeriksa skenario penggunaan untuk melihat seberapa

baik WND mendukungnya. Cukup sering, skenario penggunaan mengidentifikasi jalur melalui WND yang lebih rumit dari yang seharusnya. Analis kemudian mengerjakan ulang WND untuk menyederhanakan kemampuan antarmuka untuk mendukung skenario penggunaan, terkadang dengan membuat perubahan besar pada struktur menu, terkadang dengan menambahkan pintasan.

Desain Standar Antarmuka (Interface Standard)

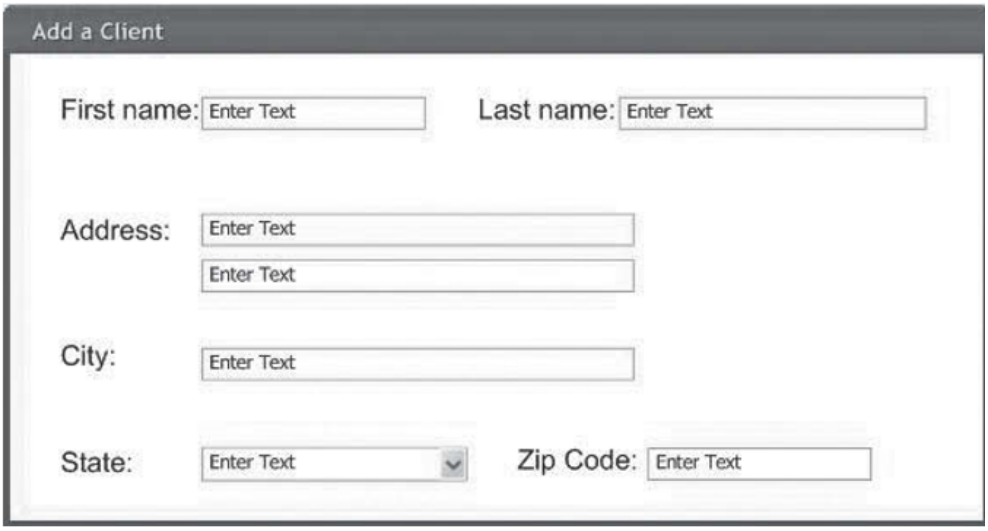
Interface Standard/Standar antarmuka adalah elemen desain dasar yang umum di seluruh layar, formulir, dan laporan individual dalam sistem. Tergantung pada aplikasinya, mungkin ada beberapa set standar antarmuka untuk berbagai bagian sistem (misalnya, satu untuk layar Web, satu untuk layar seluler, satu untuk laporan kertas, satu untuk formulir input). Misalnya, bagian dari sistem yang digunakan oleh operator entri data mungkin mencerminkan aplikasi entri data lain di perusahaan, sedangkan antarmuka Web untuk menampilkan informasi dari sistem yang sama mungkin mengikuti beberapa format Web standar. Demikian juga, setiap antarmuka individu mungkin tidak berisi semua elemen dalam standar (misalnya, layar laporan mungkin tidak memiliki kemampuan edit), dan mungkin berisi karakteristik tambahan di luar yang standar, tetapi standar berfungsi sebagai batu ujian yang memastikan antarmuka konsisten di seluruh sistem. Bagian berikut membahas beberapa area utama di mana standar antarmuka harus dipertimbangkan: metafora, objek, tindakan, ikon, dan templat.

Metafora Antarmuka (Interface Metaphor). Pertama-tama, analis harus mengembangkan metafora antarmuka dasar yang mendefinisikan bagaimana antarmuka akan bekerja. Metafora antarmuka adalah konsep dari dunia nyata yang digunakan sebagai model untuk sistem komputer. Metafora membantu pengguna memahami sistem dan memungkinkan pengguna untuk memprediksi fitur apa yang mungkin disediakan antarmuka, bahkan tanpa benar-benar menggunakan sistem. Terkadang sistem memiliki satu metafora, sedangkan dalam kasus lain ada beberapa metafora di bagian sistem yang berbeda.

Seringkali, metafora itu eksplisit. Quicken, misalnya, menggunakan metafora buku cek untuk antarmukanya, bahkan sampai pengguna mengetikkan informasi ke dalam formulir di layar yang terlihat seperti cek nyata. Dalam kasus lain, metafora itu tersirat atau tidak dinyatakan, tetapi tetap ada. Banyak sistem Windows menggunakan bentuk kertas atau tabel sebagai metafora.

Dalam beberapa kasus, metafora begitu jelas sehingga tidak memerlukan pemikiran. Misalnya, sebagian besar toko online menggunakan metafora keranjang belanja untuk menyimpan sementara barang yang sedang dipertimbangkan untuk dibeli oleh pelanggan. Dalam kasus lain, metafora sulit untuk diidentifikasi. Secara umum, lebih baik tidak memaksakan metafora yang benar-benar tidak sesuai dengan sistem, karena metafora yang tidak tepat akan membingungkan pengguna dengan mempromosikan asumsi yang salah.

Interface Tempelate/Template Antarmuka. Templat antarmuka menentukan tampilan umum semua layar dalam sistem informasi dan formulir serta laporan berbasis kertas yang digunakan. Desain template, misalnya, menentukan tata letak dasar layar (misalnya, di mana area navigasi, area status, dan area formulir/laporan akan ditempatkan) dan skema warna yang akan ditempatkan. terapan. Ini menentukan apakah jendela akan menggantikan satu sama lain di layar atau akan mengalir di atas satu sama lain. Template menentukan penempatan dan urutan standar untuk tindakan antarmuka umum (mis., Tampilan Edit File, bukan Edit Tampilan File). Singkatnya, template menyatukan elemen desain antarmuka utama lainnya: metafora, objek, tindakan, dan ikon.



Gambar 10-5 Contoh Diagram Tata Letak Windows

Interface Object/Objek Antarmuka. Template menentukan nama yang akan digunakan antarmuka untuk objek antarmuka utama, blok bangunan dasar sistem, seperti kelas. Dalam banyak kasus, nama objek bersifat langsung, seperti menyebut keranjang belanja sebagai "keranjang belanja". Dalam kasus lain, tidak sesederhana itu. Misalnya, Amazon.com menjual lebih dari sekadar buku. Dalam beberapa kasus, pengguna mungkin tidak tahu apakah dia sedang mencari buku, CD, DVD, atau unduhan Kindle. Dalam kasus tersebut, pengguna dapat menggunakan item pencarian catchall: Semua Departemen. Jika pengguna mengetahui jenis item yang ingin dibeli, pengguna dapat membatasi pencarian dengan menentukan jenis item pencarian yang lebih spesifik, seperti Aplikasi untuk Android, Buku, Kindle Store, atau Musik. Jelas, nama objek harus mudah dipahami dan membantu mempromosikan metafora antarmuka.

Secara umum, dalam kasus ketidaksepakatan antara pengguna dan analis atas nama, baik untuk objek atau tindakan (dibahas nanti), pengguna harus menang. Nama yang lebih mudah dipahami selalu mengalahkan nama yang lebih tepat atau lebih akurat.

Interface Action/Tindakan Antarmuka. Template juga menentukan gaya bahasa navigasi dan perintah (misalnya, menu) dan tata bahasa (misalnya, urutan tindakan objek; lihat bagian desain navigasi nanti dalam bab ini). Ini memberi nama untuk tindakan antarmuka yang paling umum digunakan dalam desain navigasi (misalnya, beli versus pembelian atau modifikasi versus perubahan).

Interface Icon/Ikon Antarmuka. Objek antarmuka dan tindakan dan statusnya (misalnya, dihapus atau ditarik berlebihan) dapat diwakili oleh ikon antarmuka. Ikon adalah gambar yang muncul pada tombol perintah serta dalam laporan dan formulir untuk menyoroti informasi penting. Desain ikon sangat menantang karena itu berarti mengembangkan gambar sederhana kurang dari setengah ukuran prangko yang perlu menyampaikan makna yang seringkali rumit. Pendekatan paling sederhana dan terbaik adalah dengan hanya mengadopsi ikon yang dikembangkan oleh orang lain (misalnya, halaman kosong untuk menunjukkan membuat file baru, disket untuk menunjukkan simpan). Ini memiliki keuntungan dari pengembangan ikon yang cepat, dan ikon mungkin sudah dipahami dengan baik oleh pengguna karena mereka telah melihatnya di perangkat lunak lain.

Perintah adalah tindakan yang sangat sulit direpresentasikan dengan ikon karena bergerak, tidak statis. Banyak ikon telah menjadi terkenal karena digunakan secara luas, tetapi ikon tidak dipahami dengan baik seperti yang diyakini pertama kali. Penggunaan ikon

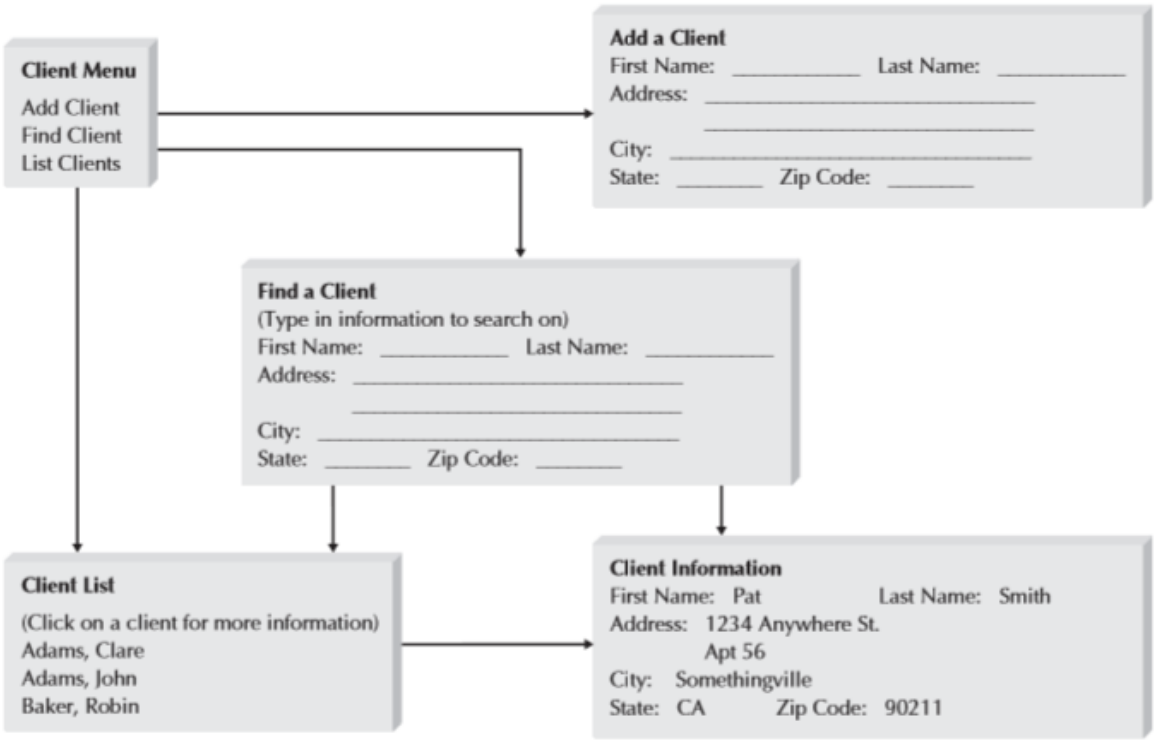
terkadang dapat menyebabkan lebih banyak kebingungan daripada wawasan. Arti ikon menjadi lebih jelas dengan penggunaan, tetapi terkadang sebuah gambar tidak bernilai bahkan satu kata; jika ragu, gunakan kata, bukan gambar.

Prototipe Desain Antarmuka

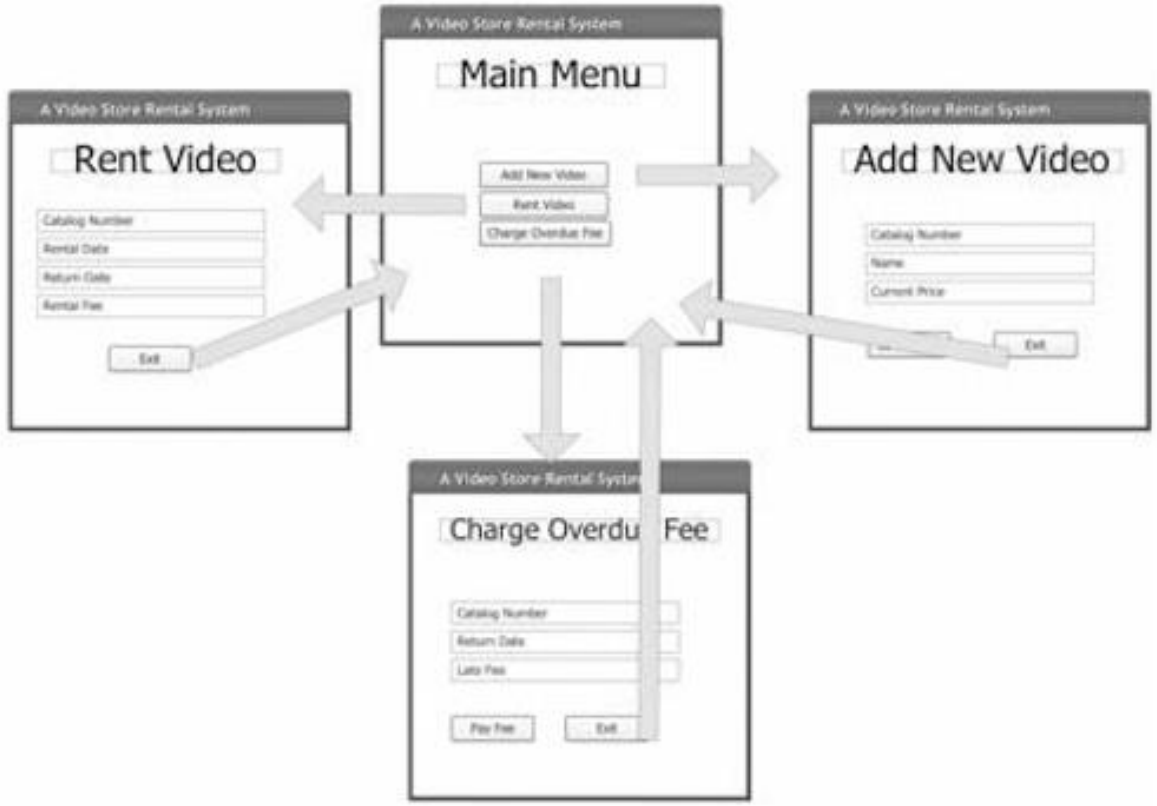
Prototipe desain antarmuka adalah tiruan atau simulasi layar komputer, formulir, atau laporan. Sebuah prototipe disiapkan untuk setiap antarmuka dalam sistem untuk menunjukkan kepada pengguna dan pemrogram bagaimana sistem akan bekerja. Di "masa lalu," prototipe desain antarmuka biasanya ditentukan pada formulir kertas yang menunjukkan apa yang akan ditampilkan di setiap bagian layar. Bentuk kertas masih digunakan sampai sekarang, tetapi semakin banyak prototipe desain antarmuka yang dibangun menggunakan alat komputer alih-alih kertas. Empat pendekatan yang paling umum untuk prototipe desain antarmuka adalah storyboard, diagram tata letak jendela, dan prototipe bahasa.

Windows Layout Diagram /Diagram Tata Letak Windows. Diagram tata letak windows hanyalah gambar yang menyerupai antarmuka pengguna sebenarnya yang akan diterima pengguna secara bertahap. Biasanya, itu dibuat menggunakan alat seperti Microsoft Visio. Dengan menggunakan alat jenis ini, perancang dapat dengan cepat menarik dan melepas komponen antarmuka pengguna ke kanvas untuk menata desain antarmuka pengguna. Misalnya, Gambar 10-5 menggambarkan diagram tata letak jendela sederhana. Meskipun tidak ada kemampuan yang dapat dieksekusi terkait dengan diagram tata letak windows, ini memungkinkan pengguna untuk dengan cepat merasakan tampilan antarmuka pengguna yang akan dikirimkan.

Storyboard. Paling sederhana, prototipe desain antarmuka adalah storyboard berbasis kertas. Storyboard menunjukkan gambar yang digambar tangan tentang tampilan layar dan bagaimana layar mengalir dari satu layar ke layar lainnya, dengan cara yang sama Storyboard untuk kartun menunjukkan bagaimana aksi akan mengalir dari satu adegan ke adegan berikutnya (lihat Gambar 10 -6). Storyboard adalah teknik yang paling sederhana karena yang mereka butuhkan hanyalah kertas (seringkali dalam flip chart) dan pena—dan seseorang dengan kemampuan artistik tertentu. Storyboard juga menggabungkan informasi navigasi dari diagram navigasi windows dan sampai tingkat tertentu informasi tata letak diagram tata letak windows. Namun, dengan alat grafis saat ini, perancang dapat bekerja secara efektif dengan sekelompok pengguna untuk merancang tampilan dan nuansa sistem yang berkembang tanpa harus benar-benar mengimplementasikan apa pun, dengan menggabungkan diagram tata letak jendela dengan diagram navigasi jendela menjadi satu kesatuan yang lebih baik. jenis diagram storyboard (lihat Gambar 10-7).



Gambar 10-6 Contoh Storyboard

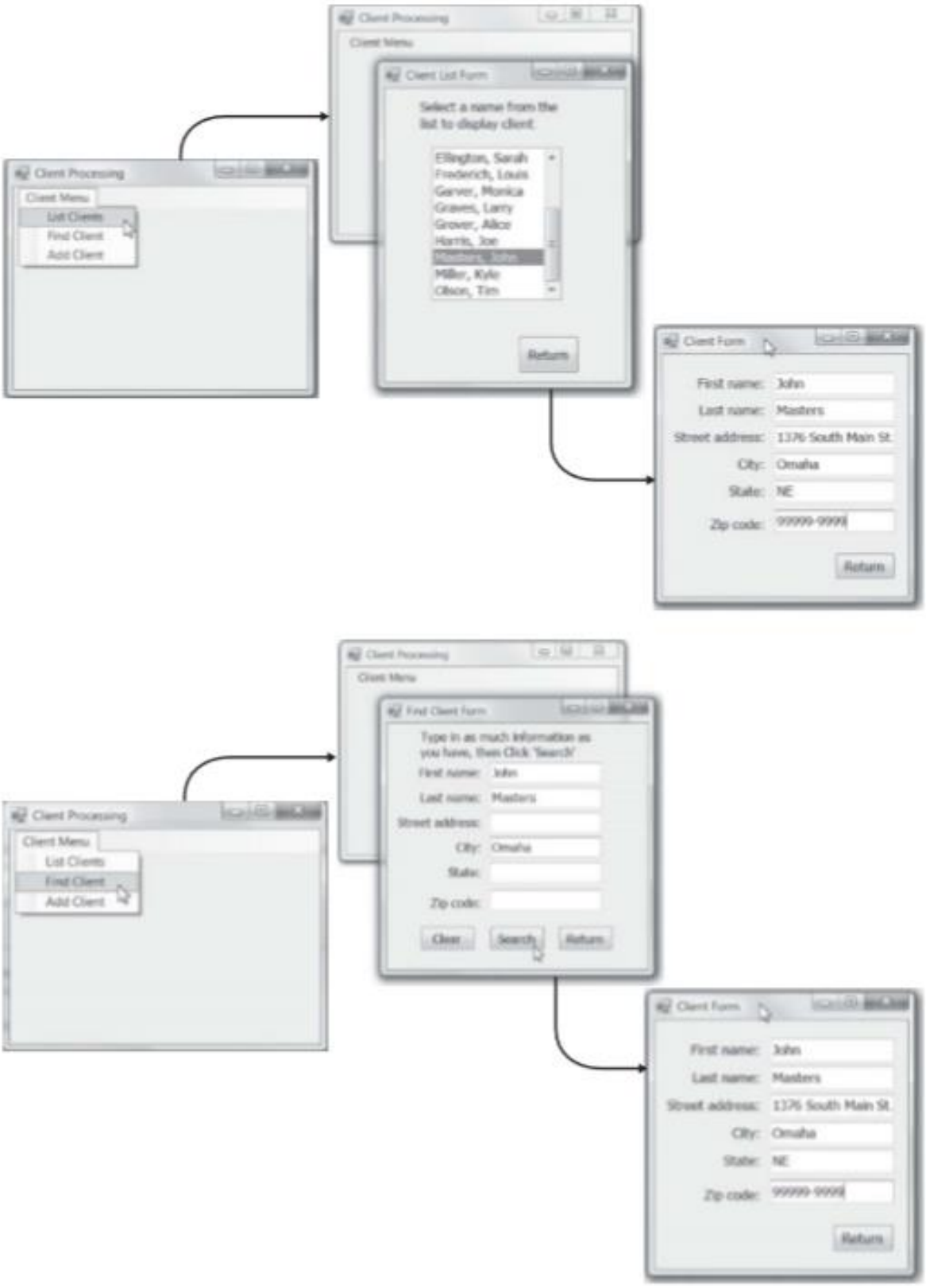


Gambar 10-7 Contoh Gabungan Navigasi dan Diagram Tata Letak Windows

Prototipe Antarmuka Pengguna. Dengan lingkungan pemrograman saat ini, seperti Visual Studio dan NetBeans, cukup mudah untuk mengembangkan prototipe yang dapat dieksekusi (lihat Gambar 10-8) dari antarmuka pengguna yang memungkinkan pengguna

untuk dapat berinteraksi dengan antarmuka pengguna dengan mengklik tombol dan memasukkan data pura-pura ke dalam formulir (tetapi karena tidak ada sistem di balik halaman, data tidak pernah diproses). Bagian berbeda dari antarmuka pengguna dihubungkan bersama sehingga saat pengguna mengklik tombol, bagian sistem yang diminta akan muncul. Prototipe yang dapat dieksekusi ini membutuhkan waktu lebih lama untuk dikembangkan daripada diagram navigasi windows, diagram tata letak windows, dan storyboard tetapi memiliki keuntungan berbeda untuk menunjukkan dengan tepat seperti apa tampilan layar. Pengguna tidak perlu menebak bentuk atau posisi elemen di layar. Namun, salah satu masalah potensial yang dapat muncul ketika mengembangkan prototipe antarmuka pengguna adalah bahwa harapan pengguna tentang kapan sistem akan selesai dapat menjadi tidak realistis. Untuk benar-benar menghubungkan prototipe ke domain masalah sehingga sistem benar-benar berfungsi bukanlah masalah sepele. Jadi, harapan pengguna perlu dikelola dengan hati-hati. Jika tidak, sistem yang memenuhi semua spesifikasinya dapat dianggap gagal.

Memilih Teknik yang Tepat. Proyek sering menggunakan kombinasi teknik prototipe desain antarmuka yang berbeda untuk berbagai bagian sistem. Storyboarding adalah yang tercepat dan paling murah tetapi memberikan detail paling sedikit. Diagram tata letak Windows memberikan lebih banyak perasaan yang akan dialami pengguna, sementara tetap cukup murah untuk dikembangkan. Prototipe antarmuka pengguna adalah pendekatan yang paling lambat, paling mahal, dan paling detail. Oleh karena itu, storyboard digunakan untuk bagian sistem di mana antarmuka dipahami dengan baik dan ketika prototipe yang lebih mahal dianggap tidak diperlukan. Namun, dalam banyak kasus mungkin sepadan dengan biaya tambahan untuk mengembangkan diagram tata letak jendela selain storyboard. Prototipe antarmuka pengguna digunakan untuk bagian sistem yang kritis, namun tidak dipahami dengan baik.



Gambar 10-8 Contoh Prototipe Antarmuka Pengguna

Evaluasi Antarmuka

Tujuan dari evaluasi antarmuka adalah untuk memahami bagaimana meningkatkan desain antarmuka sebelum sistem selesai. Sebagian besar desainer antarmuka sengaja atau tidak sengaja merancang antarmuka yang memenuhi preferensi pribadi mereka, yang mungkin

cocok atau tidak cocok dengan preferensi pengguna. Pesan utamanya, oleh karena itu, adalah agar sebanyak mungkin orang mengevaluasi antarmuka, dan semakin banyak pengguna semakin baik. Kebanyakan ahli merekomendasikan untuk melibatkan setidaknya sepuluh pengguna potensial dalam proses evaluasi.

Banyak organisasi menyimpan evaluasi antarmuka untuk langkah terakhir dalam pengembangan sistem sebelum sistem diinstal. Idealnya, bagaimanapun, evaluasi antarmuka harus dilakukan saat sistem sedang dirancang — sebelum dibangun — sehingga setiap masalah desain utama dapat diidentifikasi dan diperbaiki sebelum waktu dan biaya pemrograman dihabiskan untuk desain yang lemah. Tidak jarang sistem mengalami satu atau dua perubahan besar setelah pengguna melihat prototipe desain antarmuka pertama karena mereka mengidentifikasi masalah yang diabaikan oleh tim proyek.

Seperti halnya prototipe desain antarmuka, evaluasi antarmuka dapat mengambil banyak bentuk yang berbeda, masing-masing dengan biaya yang berbeda dan jumlah detail yang berbeda. Empat pendekatan umum adalah evaluasi heuristik, evaluasi walkthrough, evaluasi interaktif, dan pengujian kegunaan formal. Seperti halnya prototipe desain antarmuka, bagian-bagian yang berbeda dari suatu sistem dapat dievaluasi menggunakan teknik yang berbeda.

Evaluasi Heuristik. Evaluasi heuristik memeriksa antarmuka dengan membandingkannya dengan satu set heuristik atau prinsip untuk desain antarmuka. Tim proyek mengembangkan daftar periksa prinsip desain antarmuka—dari daftar di awal bab ini, misalnya, serta daftar prinsip di bagian desain navigasi, input, dan output nanti di bab ini. Setidaknya tiga anggota tim proyek kemudian secara individual bekerja melalui prototipe desain antarmuka, memeriksa setiap antarmuka untuk memastikan bahwa itu memenuhi setiap prinsip desain pada daftar periksa formal. Setelah masing-masing melalui prototipe secara terpisah, mereka bertemu sebagai sebuah tim untuk mendiskusikan evaluasi mereka dan mengidentifikasi perbaikan spesifik yang diperlukan.

Evaluasi Walkthrough. Evaluasi walkthrough desain antarmuka adalah pertemuan yang dilakukan dengan pengguna yang pada akhirnya harus mengoperasikan sistem. Tim proyek menyajikan prototipe kepada pengguna dan memandu mereka melalui berbagai bagian antarmuka. Tim proyek menunjukkan storyboard dan diagram tata letak jendela atau benar-benar mendemonstrasikan prototipe antarmuka pengguna dan menjelaskan bagaimana antarmuka akan digunakan. Pengguna mengidentifikasi perbaikan untuk setiap antarmuka yang disajikan.

Evaluasi Interaktif. Dengan evaluasi interaktif, pengguna sendiri sebenarnya bekerja dengan prototipe antarmuka pengguna dalam sesi satu orang dengan anggota tim proyek (evaluasi interaktif tidak dapat digunakan dengan storyboard atau diagram tata letak jendela). Saat pengguna bekerja dengan prototipe (seringkali dengan menggunakan skenario penggunaan, menggunakan Use-Case nyata yang dijelaskan nanti dalam bab ini, atau hanya menavigasi sesuka hati melalui sistem), dia memberi tahu anggota tim proyek apa yang dia atau dia suka dan tidak suka dan informasi atau fungsi tambahan apa yang diperlukan. Saat pengguna berinteraksi dengan prototipe, anggota tim mencatat kasus ketika dia tampak tidak yakin apa yang harus dilakukan, membuat kesalahan, atau salah menafsirkan makna komponen antarmuka. Jika pola ketidakpastian, kesalahan, atau salah tafsir terjadi kembali di beberapa pengguna yang berpartisipasi dalam evaluasi, ini merupakan indikasi yang jelas bahwa bagian-bagian antarmuka tersebut perlu diperbaiki.

Pengujian Kegunaan Formal. Pengujian kegunaan formal biasanya dilakukan dengan produk perangkat lunak komersial dan produk yang dikembangkan oleh organisasi besar yang akan digunakan secara luas melalui organisasi. Seperti namanya, ini adalah proses

yang sangat formal—hampir ilmiah—yang hanya dapat digunakan dengan prototipe bahasa (dan sistem yang telah dibangun sepenuhnya menunggu pemasangan atau pengiriman). Seperti evaluasi interaktif, pengujian kegunaan dilakukan dalam sesi satu orang di mana pengguna bekerja secara langsung dengan perangkat lunak. Namun, biasanya dilakukan di laboratorium khusus yang dilengkapi dengan kamera video dan perangkat lunak khusus yang merekam setiap penekanan tombol dan operasi mouse sehingga dapat diputar ulang untuk memahami dengan tepat apa yang dilakukan pengguna.

Pengguna diberikan serangkaian tugas khusus untuk diselesaikan (biasanya skenario penggunaan), dan setelah beberapa instruksi awal, anggota tim proyek tidak diizinkan untuk berinteraksi dengan pengguna untuk memberikan bantuan. Pengguna harus bekerja dengan perangkat lunak tanpa bantuan, yang dapat menyulitkan pengguna jika mereka bingung dengan sistem. Sangat penting bagi pengguna untuk memahami bahwa tujuannya adalah untuk menguji antarmuka, bukan kemampuan mereka, dan jika mereka tidak dapat menyelesaikan tugas, antarmuka—bukan pengguna—telah gagal dalam pengujian.

Pengujian kegunaan formal sangat mahal, karena setiap sesi satu pengguna dapat memakan waktu satu hingga dua hari untuk menganalisis tergantung pada volume detail yang dikumpulkan dalam log komputer dan video. Sesi biasanya berlangsung satu hingga dua jam. Sebagian besar pengujian kegunaan melibatkan lima hingga sepuluh pengguna, karena jika ada kurang dari lima pengguna, hasilnya terlalu bergantung pada pengguna individu tertentu yang berpartisipasi, dan lebih dari sepuluh pengguna seringkali terlalu mahal untuk dibenarkan (kecuali pengembang perangkat lunak komersial besar terlibat).

Pendekatan Akal Sehat untuk Desain Antarmuka Pengguna

Ketika Anda mempertimbangkan semua materi di atas, membuat desain antarmuka pengguna yang efektif bisa menjadi tugas yang menakutkan dan sangat memakan waktu. Sebuah buku menarik oleh Steve Krug, bagaimanapun, memberi kita seperangkat prinsip panduan untuk kegunaan Web. Di bagian ini, kami mengadaptasi prinsipnya ke desain antarmuka pengguna umum.

Pertama, pengguna tidak perlu memikirkan cara menavigasi antarmuka pengguna. Seperti yang dikatakan Krug, "Jangan membuatku berpikir." Secara kognitif, setiap kali pengguna harus berhenti dan mencari cara untuk menggunakan antarmuka pengguna, pembuat antarmuka pengguna telah gagal. Itu mungkin tampak agak kasar, tetapi itu benar. Dari sudut pandang pengguna, antarmuka pengguna adalah sistem. Jika pengembang telah melakukan pekerjaan rumah mereka, antarmuka pengguna harus intuitif untuk digunakan. Dari perspektif praktis, kita harus mempelajari bagaimana pengguna benar-benar menggunakan sistem. Berdasarkan pengamatan Krug terhadap pengguna, ia menemukan bahwa pengguna tidak membaca halaman Web; sebaliknya, mereka cenderung memindainya. Sebagai pedoman umum desain antarmuka pengguna, kami menyarankan agar Anda memudahkan pengguna untuk mengidentifikasi berbagai bagian antarmuka pengguna sehingga mereka cukup memindai layar untuk melihat bagian antarmuka yang sesuai dengan masalah yang mereka selesaikan . Mengingat kecenderungan pengguna untuk hanya memindai antarmuka pengguna, Krug menyarankan agar kita mempertimbangkan untuk mempelajari papan reklame untuk mendapatkan inspirasi. Billboard dirancang untuk "dibaca" pada kecepatan 70 mph saat Anda berkendara di jalan raya. Jelas, informasi yang paling relevan harus menarik perhatian Anda agar iklan billboard berfungsi. Dia menyarankan bahwa kita harus menggunakan seperangkat konvensi yang kita kenal. Misalnya, ketika melihat sebuah surat kabar Anda tahu bahwa surat kabar itu disusun menjadi bagian-bagian yang berbeda. Dalam kasus Wall Street Journal, Anda tahu bahwa halaman depan bertindak sebagai

indeks ke seluruh kertas. Akibatnya, kita harus mencari konvensi yang dapat kita terapkan untuk membantu pengguna.

Kedua, ia menyarankan bahwa jumlah klik yang harus dilakukan pengguna untuk menyelesaikan tugas agak tidak relevan. Alih-alih, berdasarkan prinsip panduan pertamanya, yang penting adalah mendesain antarmuka pengguna sedemikian rupa sehingga pilihan (klik) yang akan dibuat tidak ambigu. Membuat banyak pilihan yang jelas jauh lebih cepat dan lebih mudah daripada beberapa pilihan yang tidak jelas dan ambigu. Akibatnya, jangan khawatir tentang jumlah layar yang harus dilalui pengguna. Namun, seperti aturan lainnya, ini bisa dianggap ekstrem. Terlalu banyak klik masih terlalu banyak klik. Tujuan keseluruhannya adalah untuk meminimalkan upaya pengguna. Cukup fokus untuk memudahkan pengguna menyelesaikan tugas.

Ketiga, meminimalkan jumlah kata di layar. Mengingat bahwa pengguna memindai layar untuk menemukan apa yang mereka cari, buat lebih mudah dengan tidak mengacaukan layar dengan banyak kebisingan. Dia menyarankan bahwa dalam kasus antarmuka Web, 50 persen hingga 75 persen kata dapat dihilangkan tanpa kehilangan informasi apa pun yang ada di layar. Jelas, ini mungkin agak ekstrem, tetapi ini menunjukkan bahwa mengikuti prinsip KISS sangat penting ketika merancang antarmuka pengguna yang efektif.

10.5 DESAIN NAVIGASI

Komponen navigasi antarmuka memungkinkan pengguna untuk memasukkan perintah untuk menavigasi melalui sistem dan melakukan tindakan untuk memasukkan dan meninjau informasi yang dikandungnya. Komponen navigasi juga menyajikan pesan kepada pengguna tentang keberhasilan atau kegagalan tindakannya. Tujuan dari sistem navigasi adalah untuk membuat sistem sesederhana mungkin untuk digunakan. Komponen navigasi yang baik adalah komponen yang tidak pernah benar-benar diperhatikan oleh pengguna. Ini hanya berfungsi seperti yang diharapkan pengguna, dan dengan demikian pengguna tidak terlalu memikirkannya. Dengan kata lain, ingatlah tiga prinsip panduan Krug saat Anda mengerjakan tiga bagian teks berikutnya.

Prinsip dasar

Salah satu hal tersulit tentang menggunakan sistem komputer adalah mempelajari cara memanipulasi kontrol navigasi untuk membuat sistem melakukan apa yang Anda inginkan. Analis harus berasumsi bahwa pengguna belum membaca manual, belum menghadiri pelatihan, dan tidak memiliki bantuan eksternal yang siap sedia. Semua kontrol harus jelas dan dapat dipahami serta ditempatkan di lokasi yang intuitif di layar. Idealnya, kontrol harus mengantisipasi apa yang akan dilakukan pengguna dan menyederhanakan usahanya. Misalnya, banyak program pengaturan yang dirancang sedemikian rupa sehingga untuk penginstalan biasa, pengguna dapat terus menekan tombol Berikutnya.

Mencegah Kesalahan. Prinsip pertama dalam merancang kontrol navigasi adalah untuk mencegah pengguna membuat kesalahan. Sebuah kesalahan menghabiskan waktu dan menyebabkan frustrasi. Lebih buruk lagi, serangkaian kesalahan dapat menyebabkan pengguna membuang sistem. Kesalahan dapat dikurangi dengan memberi label perintah dan tindakan secara tepat dan dengan membatasi pilihan. Terlalu banyak pilihan dapat membingungkan pengguna, terutama jika pilihannya serupa dan sulit untuk dijelaskan dalam ruang pendek yang tersedia di layar. Ketika ada banyak pilihan serupa pada menu, pertimbangkan untuk membuat level menu kedua atau serangkaian opsi untuk perintah dasar.

Jangan pernah menampilkan perintah yang tidak dapat digunakan. Banyak aplikasi Windows menghapus perintah yang tidak dapat digunakan; mereka ditampilkan pada menu

pull-down dalam font berwarna sangat terang, tetapi mereka tidak dapat dipilih. Ini menunjukkan bahwa mereka tersedia tetapi tidak dapat digunakan dalam konteks saat ini. Itu juga menyimpan semua item menu di tempat yang sama.

Ketika pengguna akan melakukan fungsi kritis yang sulit atau tidak mungkin untuk dibatalkan (misalnya, menghapus file), penting untuk mengonfirmasi tindakan tersebut dengan pengguna (dan pastikan pemilihan tidak dilakukan karena kesalahan). Meminta pengguna menanggapi pesan konfirmasi, yang menjelaskan apa yang diminta pengguna dan meminta pengguna untuk mengonfirmasi bahwa tindakan ini benar, biasanya dilakukan.

Sederhanakan Pemulihan dari Kesalahan. Tidak peduli apa yang dilakukan perancang sistem, pengguna akan membuat kesalahan. Sistem harus membuatnya semudah mungkin untuk memperbaiki kesalahan ini. Saya setuju, sistem memiliki tombol Undo yang membuat kesalahan mudah ditimpa; namun, menulis perangkat lunak untuk tombol tersebut bisa sangat rumit.

Gunakan Tata Bahasa yang Konsisten. Urutan Salah satu keputusan yang paling mendasar adalah urutan tata bahasa. Sebagian besar perintah mengharuskan pengguna untuk menentukan objek (mis., File, record, Word), dan tindakan yang akan dilakukan pada objek itu (mis., Salin, hapus). Antarmuka dapat meminta pengguna untuk terlebih dahulu memilih objek dan kemudian tindakan (urutan objek-tindakan) atau pertama-tama memilih tindakan dan kemudian objek (urutan tindakan-objek). Sebagian besar aplikasi Windows menggunakan urutan tata bahasa tindakan objek (mis., Pikirkan tentang menyalin blok teks di pengolah kata Anda).

Urutan tata bahasa harus konsisten di seluruh sistem, baik di tingkat elemen data maupun di tingkat menu secara keseluruhan. Para ahli berdebat tentang keuntungan dari satu pendekatan di atas yang lain, tetapi karena sebagian besar pengguna akrab dengan urutan tindakan objek, sebagian besar sistem saat ini dirancang menggunakan pendekatan itu.

Jenis Kontrol Navigasi

Ada dua perangkat keras tradisional yang dapat digunakan untuk mengontrol antarmuka pengguna: keyboard dan perangkat penunjuk seperti mouse, trackball, atau layar sentuh. Saat ini, tergantung pada perangkat keras yang digunakan, sistem pengenalan suara juga dapat digunakan untuk mengontrol antarmuka pengguna. Ada tiga pendekatan perangkat lunak dasar untuk mendefinisikan perintah pengguna: bahasa, menu, dan manipulasi langsung.

Bahasa. Dengan bahasa perintah, pengguna memasukkan perintah menggunakan bahasa khusus yang dikembangkan untuk sistem komputer (misalnya, UNIX dan SQL keduanya menggunakan bahasa perintah). Bahasa perintah terkadang memberikan fleksibilitas yang lebih besar daripada pendekatan lain karena pengguna dapat menggabungkan elemen bahasa dengan cara yang tidak ditentukan sebelumnya oleh pengembang. Namun, mereka memberikan beban yang lebih besar pada pengguna karena pengguna harus mempelajari sintaks dan mengetik perintah daripada memilih dari sejumlah pilihan yang terbatas dan terdefinisi dengan baik. Sistem saat ini menggunakan bahasa perintah dengan hemat, kecuali dalam kasus di mana ada sejumlah besar kombinasi perintah yang membuatnya tidak praktis untuk mencoba membangun semua kombinasi ke dalam menu (misalnya, kueri SQL untuk database).

Antarmuka bahasa alami dirancang untuk memahami bahasa pengguna sendiri (misalnya, Inggris, Prancis, Indonesia, Korea, China, Aran, dan Spanyol). Antarmuka ini mencoba untuk menafsirkan apa yang dimaksud pengguna, dan seringkali mereka menyajikan kembali kepada pengguna daftar interpretasi yang dapat dipilih. Contoh penggunaan bahasa

alami adalah mesin pencari Google. Mesin pencari Google memungkinkan pengguna untuk menggunakan teks bentuk bebas untuk mencari di Web untuk topik yang menarik.

Menu. Jenis sistem navigasi yang paling umum saat ini adalah menu. Menu menyajikan pengguna dengan daftar pilihan, yang masing-masing dapat dipilih. Menu lebih mudah dipelajari daripada bahasa karena sejumlah perintah yang tersedia disajikan kepada pengguna secara terorganisir. Mengklik item dengan alat penunjuk atau menekan tombol yang cocok dengan pilihan menu (misalnya, tombol fungsi) membutuhkan sedikit usaha. Oleh karena itu, menu biasanya lebih disukai daripada bahasa.

Menu perlu dirancang dengan hati-hati karena submenu di belakang menu utama disembunyikan dari pengguna sampai mereka mengklik item menu. Lebih baik membuat menu yang luas dan dangkal (yaitu, setiap menu berisi banyak item dengan hanya satu atau dua lapisan menu) daripada sempit dan dalam (yaitu, setiap menu hanya berisi beberapa item, tetapi masing-masing mengarah ke tiga atau lebih lapisan dari menu). Menu yang luas dan dangkal menyajikan informasi paling banyak kepada pengguna pada awalnya sehingga dia dapat melihat banyak opsi dan hanya memerlukan beberapa klik mouse atau penekanan tombol untuk melakukan suatu tindakan. Menu yang sempit dan dalam membuat pengguna mencari item yang tersembunyi di balik item menu dan membutuhkan lebih banyak klik atau penekanan tombol untuk melakukan suatu tindakan.

Penelitian menunjukkan bahwa di dunia yang ideal, setiap menu tidak boleh berisi lebih dari delapan item, dan tidak boleh lebih dari dua klik mouse atau penekanan tombol dari menu mana pun untuk melakukan tindakan (atau tiga dari menu utama yang memulai sistem) . Namun, analis terkadang harus melanggar pedoman ini dalam desain sistem yang kompleks dengan mengelompokkan item menu yang dipisahkan oleh garis horizontal. Seringkali item menu memiliki tombol pintas yang memungkinkan pengguna berpengalaman untuk dengan cepat menjalankan perintah dengan penekanan tombol sebagai pengganti pilihan menu (misalnya, pada mesin Windows, di banyak aplikasi, Ctrl-F cenderung menjalankan perintah Temukan; pada Mac, Anda gunakan Command-F sebagai gantinya).

Menu harus disatukan seperti item sehingga pengguna secara intuitif dapat menebak isi setiap menu. Kebanyakan desainer merekomendasikan pengelompokan item menu dengan objek antarmuka (misalnya, pelanggan, pesanan pembelian, persediaan) daripada dengan tindakan antarmuka (misalnya, baru, update, format), sehingga semua tindakan yang berkaitan dengan satu objek berada dalam satu menu, semua tindakan untuk objek lain berada di menu yang berbeda, dan seterusnya. Namun, ini sangat tergantung pada antarmuka tertentu. Beberapa jenis menu yang lebih umum termasuk bilah menu, menu drop-down, menu pop-up, menu tab, bilah alat, dan peta gambar (lihat Gambar 10-9).

Jenis Menu	Kapan digunakan	Catatan
Menu bar		
Daftar perintah di bagian atas layar; selalu di layar	Menu utama untuk sistem	Gunakan organisasi yang sama dengan sistem operasi dan paket lainnya (mis., File, Edit, View). Item menu selalu satu kata, tidak pernah dua. Item menu mengarah ke menu lain daripada melakukan tindakan. Jangan pernah izinkan pengguna memilih tindakan yang tidak dapat mereka lakukan (sebagai gantinya, gunakan item berwarna abu-abu).

Drop down Menu		
Menu yang turun tepat di bawah menu lain; menghilang setelah satu kali digunakan	Menu tingkat kedua, seringkali dari bilah menu	Item menu seringkali terdiri dari beberapa kata. Hindari singkatan. Item menu melakukan tindakan atau mengarah ke menu drop-down berjenjang, menu pop-up, atau menu tab lainnya.
Pop-up Menu		
Menu yang muncul dan mengambang di atas layar; menghilang setelah satu kali digunakan	Sebagai jalan pintas ke perintah untuk pengguna berpengalaman	Menu pop-up sering (tidak selalu) dipanggil dengan klik kanan di sistem berbasis Windows. Menu-menu ini sering diabaikan oleh pengguna pemula, jadi biasanya mereka harus menduplikasi fungsi yang disediakan di menu lain
Tab menu		
Menu multihalaman dengan satu tab untuk setiap halaman yang muncul dan mengapung di atas layar; tetap di layar sampai ditutup	Ketika pengguna perlu mengubah beberapa pengaturan atau melakukan beberapa perintah terkait	Item menu harus pendek agar pas dengan label tab. Hindari lebih dari satu baris tab, karena mengklik tab untuk membukanya dapat mengubah urutan tab dan hampir tidak ada kasus lain yang memilih dari menu mengatur ulang menu itu sendiri.
Tool bar		
Menu tombol (seringkali dengan ikon) yang tetap ada di layar hingga ditutup	Sebagai jalan pintas ke perintah untuk pengguna berpengalaman	Semua tombol pada bilah alat yang sama harus berukuran sama. Jika ukuran label sangat bervariasi, gunakan dua ukuran berbeda (kecil dan besar). Tombol dengan ikon harus memiliki tip alat, area yang menampilkan frasa teks yang menjelaskan tombol saat pengguna menjeda mouse di atasnya.
Image Map		
Gambar grafis di mana area tertentu terkait dengan tindakan atau menu lainnya	Hanya ketika gambar grafis menambahkan makna ke menu	Gambar harus menyampaikan makna untuk menunjukkan bagian mana yang melakukan tindakan saat diklik. Kiat alat dapat membantu.

Gambar 10-9 Jenis Menu

Manipulasi Langsung. Dengan manipulasi langsung, pengguna memasukkan perintah dengan bekerja secara langsung dengan objek antarmuka. Misalnya, pengguna dapat mengubah ukuran objek di Microsoft PowerPoint dengan mengkliknya dan memindahkan sisinya, atau mereka dapat memindahkan file di Windows Explorer dengan menyeret nama file dari satu folder ke folder lain. Manipulasi langsung bisa sederhana, tetapi memiliki dua masalah. Pertama, pengguna yang akrab dengan antarmuka berbasis bahasa atau menu tidak selalu mengharapkannya. Kedua, tidak semua perintah bersifat intuitif. [Bagaimana Anda menyalin (tidak memindahkan) file di Windows Explorer? Di Macintosh, mengapa memindahkan folder ke tempat sampah menghapus file jika ada di hard disk, tetapi mengeluarkan DVD jika file ada di DVD?]

Pesan

Pesan adalah cara sistem merespons pengguna dan memberi tahu dia tentang status interaksi. Ada banyak jenis pesan yang berbeda, seperti pesan kesalahan, pesan konfirmasi, pesan pengakuan, pesan tunda, dan pesan bantuan (lihat Gambar 10-10). Secara umum, pesan harus jelas, ringkas, dan lengkap, yang terkadang merupakan tujuan yang saling bertentangan. Semua pesan harus benar secara tata bahasa dan bebas dari jargon dan singkatan (kecuali jika itu adalah jargon dan singkatan pengguna). Hindari negatif karena dapat membingungkan (misalnya, ganti Apakah Anda yakin tidak ingin melanjutkan? dengan Apakah Anda ingin berhenti?). Demikian juga, hindari humor, karena akan cepat habis setelah pesan yang sama muncul puluhan kali.

Jenis Pesan	Kaoan Digunakan	Catatan
Pesan Eror		
Memberi tahu pengguna bahwa dia telah mencoba melakukan sesuatu yang tidak dapat ditanggapi oleh sistem	Ketika Pengguna melakukan sesuatu yang tidak diizinkan atau tidak mungkin	• Selalu jelaskan alasannya dan sarankan tindakan korektif. • Biasanya, pesan kesalahan disertai dengan bunyi bip, tetapi banyak aplikasi sekarang menghilangkannya atau mengizinkan pengguna untuk menghapusnya.
pesan konfirmasi		
Meminta pengguna untuk mengonfirmasi bahwa mereka benar-benar ingin melakukan tindakan yang telah mereka pilih	Saat pengguna memilih pilihan yang berpotensi berbahaya, seperti menghapus file	• Selalu jelaskan penyebabnya dan sarankan tindakan yang mungkin dilakukan. • Sering menyertakan beberapa pilihan selain OK dan batal.
Pesan pengakuan		
Memberi tahu pengguna bahwa sistem telah menyelesaikan apa yang diminta untuk dilakukan	Jarang atau tidak pernah. Pengguna dengan cepat menjadi terganggu dengan semua klik mouse yang tidak perlu	• Pesan pengakuan biasanya Disertakan karena pengguna pemula sering kali ingin diyakinkan bahwa suatu tindakan telah terjadi. • Pendekatan terbaik adalah memberikan informasi pengakuan tanpa pesan terpisah yang harus diklik oleh pengguna. • Misalnya, jika pengguna melihat item dalam daftar dan menambahkannya, maka daftar yang diperbarui di layar yang menunjukkan item yang ditambahkan sudah cukup sebagai pengakuan.
Pesan tunda		
Memberi tahu pengguna bahwa sistem komputer berfungsi dengan baik	Ketika suatu aktivitas membutuhkan waktu lebih dari tujuh detik	• Harus mengizinkan pengguna untuk membatalkan operasi jika dia tidak ingin menunggu penyelesaiannya. • Harus memberikan beberapa indikasi berapa lama penundaan akan berlangsung.

Pesan bantuan		
Memberikan informasi tambahan tentang sistem dan komponennya	Di semua sistem	• Informasi bantuan diatur menurut daftar isi dan/atau pencarian kata kunci. • Bantuan peka konteks memberikan informasi yang bergantung pada apa yang dilakukan pengguna saat bantuan diminta. • Pesan bantuan dan dokumentasi online dibahas di Bab 12.

Gambar 10-10 Jenis Pesan

Pesan harus meminta pengguna untuk mengakuinya (dengan mengklik, misalnya), daripada ditampilkan selama beberapa detik dan kemudian menghilang. Pengecualian adalah pesan yang memberi tahu pengguna tentang penundaan dalam pemrosesan, yang akan hilang setelah penundaan berlalu. Secara umum, pesan adalah teks, tetapi terkadang, ikon standar digunakan. Misalnya, Windows menampilkan jam pasir saat sistem sedang sibuk. Semua pesan harus dibuat dengan hati-hati, tetapi pesan kesalahan dan bantuan memerlukan perhatian khusus. Pesan (dan khususnya pesan kesalahan) harus selalu menjelaskan masalah dengan bahasa yang sopan dan ringkas (misalnya, kesalahan yang dilakukan pengguna) dan menjelaskan tindakan korektif sejelas dan sejelas mungkin sehingga pengguna tahu persis apa yang perlu dilakukan. Dalam kasus kesalahan yang rumit, pesan kesalahan harus menampilkan apa yang dimasukkan pengguna, menyarankan kemungkinan penyebab kesalahan, dan mengusulkan kemungkinan tanggapan pengguna. Jika ragu, berikan lebih banyak informasi daripada yang dibutuhkan pengguna atau kemampuan untuk mendapatkan informasi tambahan. Pesan kesalahan harus memberikan nomor pesan. Nomor pesan tidak ditujukan untuk pengguna, tetapi kehadirannya mempermudah meja bantuan dan saluran dukungan pelanggan untuk mengidentifikasi masalah dan membantu pengguna karena banyak pesan menggunakan kata-kata yang serupa.

Dokumentasi Desain Navigasi

Desain navigasi untuk suatu sistem dilakukan melalui penggunaan WND dan Use-Case nyata. Use-Case nyata diturunkan dari Use-Case esensial (lihat Bab 4), skenario penggunaan, dan WND. Ingat bahwa Use-Case esensial adalah kasus yang hanya menjelaskan masalah esensial minimum yang diperlukan untuk memahami fungsionalitas yang diperlukan. Use-Case nyata menggambarkan serangkaian langkah spesifik yang dilakukan pengguna untuk menggunakan bagian tertentu dari suatu sistem. Use-Case nyata bergantung pada implementasi (yaitu, deskripsi terperinci tentang cara menggunakan sistem setelah diimplementasikan).

Untuk mengembangkan use case esensial menjadi use case nyata, dua perubahan harus dilakukan. Pertama, tipe use case harus diubah dari essential menjadi real. Kedua, semua peristiwa harus ditentukan dalam hal antarmuka pengguna yang sebenarnya. Dan, mengingat kekhasan platform yang berbeda, misalnya, desktop, tablet, dan smartphone, Use-Case nyata perlu dikembangkan untuk setiap platform tempat Use-Case diterapkan. Oleh karena itu, aliran normal peristiwa, sub-aliran, dan aliran alternatif/luar biasa harus dimodifikasi. Alur normal dari kejadian, sub-aliran, dan aliran alternatif/luar biasa untuk Use-Case nyata yang terkait dengan prototipe antarmuka pengguna storyboard yang diberikan pada Gambar 10-6 ditunjukkan pada Gambar 10-11. Misalnya, langkah 2 dari aliran peristiwa normal menyatakan bahwa "Sistem menyediakan Staf Penjualan dengan Menu Utama untuk Sistem,"

yang memungkinkan Perwakilan Penjualan berinteraksi dengan aspek Pemeliharaan Daftar Klien dari sistem.

10.6 INPUT DESAIN

Input memfasilitasi masuknya data ke dalam sistem komputer, baik data yang sangat terstruktur, seperti informasi pesanan (misalnya, nomor barang, jumlah, biaya) atau informasi tidak terstruktur (misalnya, komentar). Desain input berarti mendesain layar yang digunakan untuk memasukkan informasi serta bentuk apa pun yang digunakan pengguna untuk menulis atau mengetik informasi (misalnya, kartu waktu, klaim pengeluaran).

Prinsip Dasar

Tujuan dari mekanisme input adalah untuk secara sederhana dan mudah menangkap informasi yang akurat untuk sistem. Prinsip dasar untuk desain input mencerminkan sifat input (baik batch atau online) dan cara untuk menyederhanakan pengumpulannya.

Nama Use Case : Maintain Client List	ID : 12	Level Kesulitan : Tinggi
Primary Actor: Sales Rep	Use-Case Type: Detail, Real	
Stakeholders and Interests: Sales Rep - wants to add, f ind, or list clients		
Deskripsi Singkat: Use-Case ini menjelaskan bagaimana perwakilan penjualan dapat mencari dan memelihara daftar klien.		
Pemicu: Pasien menelepon dan meminta janji baru atau meminta untuk membatalkan atau mengubah janji yang sudah ada. Type : External		
Relationships: Association: Sales Rep Include: Extend: Generalization:		
Alur Peristiwa Normal: 11. Perwakilan Penjualan memulai sistem. 12. Sistem menyediakan Sales Rep dengan Menu Utama untuk Sistem. 13. Sistem menanyakan Sales Rep apakah dia ingin menambahkan klien. Temukan Klien yang ada, atau Daftar semua klien yang ada. Jika Sales Rep ingin menambahkan klien, dia mengklik Link Tambah Klien dan menjalankan S-1: Klien Baru. Jika Perwakilan Penjualan ingin menemukan klien, dia mengklik Tautan Temukan Klien dan jalankan S-2: Temukan Klien. Jika Sales Rep ingin membuat daftar semua klien, dia mengklik Link Daftar Klien dan menjalankan S-3: Daftar Klien.		

14. Sistem mengembalikan Sales Rep ke Menu Utama Sistem.

Subflow:

S-1: Klien Baru

1. Sistem meminta Perwakilan Penjualan untuk informasi yang relevan.
2. Perwakilan Penjualan mengetikkan informasi yang relevan ke dalam Formulir
3. Perwakilan Penjualan mengirimkan informasi ke Sistem.

S-2: Temukan Klien

1. Sistem meminta Sales Rep untuk mencari informasi.
2. Rep Penjualan mengetik informasi pencarian ke dalam Formulir
3. Perwakilan Penjualan mengirimkan informasi ke Sistem.

   Jika Sistem menemukan satu Klien yang memenuhi informasi pencarian, Sistem menghasilkan laporan Informasi Klien dan mengembalikan Sales Rep ke Menu Utama Sistem

   Lain

   Jika Sistem menemukan daftar Klien yang memenuhi informasi pencarian. Sistem menjalankan S-3: Daftar Klien.

S-3: Daftar Klien

1. Jika Subflow ini dijalankan dari Langkah 3

   Sistem membuat Daftar Semua klien

   Lain

   Sistem membuat Daftar klien yang cocok dengan kriteria pencarian S-2: Temukan Klien.
2. Perwakilan Penjualan memilih klien.
3. Sistem menghasilkan laporan Informasi Klien.

Aliran Alternatif/Luar Biasa:

S-2 4a. Sistem menghasilkan Pesan Kesalahan

Gambar 10-11 Contoh Use-Case Nyata

Pemrosesan Online versus Batch. Ada dua format umum untuk memasukkan input ke dalam sistem komputer: pemrosesan online dan pemrosesan batch. Dengan pemrosesan online (kadang-kadang disebut pemrosesan transaksi), setiap item input (misalnya, pesanan pelanggan, pesanan pembelian) dimasukkan ke dalam sistem secara individual, biasanya pada saat yang sama dengan peristiwa atau transaksi yang meminta input. Misalnya, ketika Anda memeriksa buku dari perpustakaan, membeli barang di toko, atau membuat reservasi penerbangan, sistem komputer yang mendukung proses tersebut menggunakan pemrosesan online untuk segera mencatat transaksi di database yang sesuai. Pemrosesan online paling sering digunakan ketika penting untuk memiliki informasi waktu nyata tentang proses bisnis. Misalnya, saat Anda memesan kursi maskapai, kursi tersebut tidak lagi tersedia untuk digunakan orang lain.

Dengan pemrosesan batch, semua input yang dikumpulkan selama beberapa periode waktu dikumpulkan bersama dan dimasukkan ke dalam sistem pada satu waktu dalam satu batch. Beberapa proses bisnis secara alami menghasilkan informasi dalam batch. Misalnya, sebagian besar penggajian per jam dilakukan menggunakan pemrosesan batch karena kartu waktu dikumpulkan bersama dalam batch dan diproses sekaligus. Pemrosesan batch juga digunakan untuk sistem pemrosesan transaksi yang tidak memerlukan informasi waktu nyata. Misalnya, sebagian besar toko mengirimkan informasi penjualan ke kantor distrik sehingga inventaris pengganti baru dapat dipesan. Informasi ini dapat dikirim secara real time seperti

yang ditangkap di toko sehingga kantor distrik mengetahui dalam satu atau dua detik bahwa suatu produk telah terjual. Jika toko tidak memerlukan data real-time terbaru ini, mereka akan mengumpulkan data penjualan sepanjang hari dan mengirimkannya setiap malam secara berkelompok ke kantor distrik. Pengelompokan ini menyederhanakan proses komunikasi data dan seringkali menghemat biaya komunikasi, tetapi ini berarti bahwa persediaan tidak akurat secara real time melainkan akurat hanya pada akhir hari setelah kumpulan diproses.

Capture Data di Sumbernya. Mungkin prinsip terpenting dari desain input adalah menangkap data dalam format elektronik pada sumber aslinya atau sedekat mungkin dengan sumber aslinya. Pada hari-hari awal komputasi, sistem komputer menggantikan sistem manual tradisional yang beroperasi pada formulir kertas. Karena proses bisnis ini diotomatisasi, banyak bentuk kertas asli tetap ada, baik karena tidak ada yang berpikir untuk menggantinya atau karena terlalu mahal untuk melakukannya. Sebaliknya, proses bisnis terus berisi formulir manual yang dibawa ke pusat komputer dalam batch untuk diketik ke dalam sistem komputer oleh operator entri data.

Banyak proses bisnis masih beroperasi dengan cara ini sampai sekarang. Sebagai contoh, sebagian besar organisasi memiliki formulir klaim pengeluaran yang dilengkapi dengan tangan dan diserahkan ke departemen akuntansi, yang menyetujuinya dan memasukkannya ke dalam sistem secara berkelompok. Ada tiga masalah dengan pendekatan ini. Pertama, mahal karena menduplikasi pekerjaan (formulir diisi dua kali, sekali dengan tangan, sekali dengan keyboard). Kedua, meningkatkan waktu pemrosesan karena formulir kertas harus dipindahkan secara fisik melalui proses. Ketiga, meningkatkan biaya dan kemungkinan kesalahan, karena memisahkan entri dari pemrosesan informasi; seseorang mungkin salah membaca tulisan tangan pada formulir input, data mungkin dimasukkan salah, atau input asli dapat berisi kesalahan yang membuat informasi menjadi tidak valid.

Sebagian besar sistem pemrosesan transaksi saat ini dirancang untuk menangkap data pada sumbernya. Otomatisasi data sumber mengacu pada penggunaan perangkat keras khusus untuk secara otomatis menangkap data tanpa mengharuskan siapa pun untuk mengetiknya. Toko biasanya menggunakan pembaca kode batang yang secara otomatis memindai produk dan memasukkan data langsung ke sistem komputer. Tidak ada format perantara seperti formulir kertas yang digunakan. Teknologi serupa termasuk pengenalan karakter optik, yang dapat membaca angka dan teks yang dicetak (misalnya, pada cek), pembaca strip magnetik, yang dapat membaca informasi yang dikodekan pada strip magnetik (misalnya, kartu kredit), dan kartu pintar, yang berisi mikroprosesor, chip memori , dan baterai (seperti kalkulator seukuran kartu kredit). Selain mengurangi waktu dan biaya entri data, sistem ini mengurangi kesalahan karena mereka jauh lebih kecil kemungkinannya untuk menangkap data secara tidak benar. Saat ini, komputer portabel dan pemindai memungkinkan data diambil dari sumbernya bahkan dalam pengaturan seluler (misalnya, pengiriman kurir udara, penggunaan mobil sewaan).

Sistem otomatis ini tidak mampu mengumpulkan banyak informasi, jadi pilihan terbaik berikutnya adalah mengambil data segera dari sumbernya menggunakan operator entri yang terlatih. Banyak reservasi maskapai dan hotel, aplikasi pinjaman, dan pesanan katalog dicatat langsung ke dalam sistem komputer, sementara pelanggan memberikan jawaban atas pertanyaan kepada operator. Beberapa sistem menghilangkan operator sama sekali dan memungkinkan pengguna untuk memasukkan data mereka sendiri. Misalnya, banyak universitas tidak lagi menerima aplikasi berbasis kertas untuk penerimaan; semua aplikasi diketik oleh siswa ke dalam bentuk elektronik.

Formulir untuk menangkap informasi (di layar, di atas kertas, dll.) harus mendukung sumber data. Artinya, urutan informasi pada formulir harus sesuai dengan aliran informasi

alami dari sumber data, dan formulir entri data harus sesuai dengan formulir kertas yang digunakan untuk menangkap data pada awalnya.

Minimalkan Keystroke. Prinsip penting lainnya adalah meminimalkan penekanan tombol. Penekanan tombol menghabiskan waktu dan uang, baik dilakukan oleh pelanggan, pengguna, atau operator entri data yang terlatih. Sistem tidak boleh meminta informasi yang dapat diperoleh dengan cara lain (misalnya, dengan mengambilnya dari database atau dengan melakukan perhitungan). Demikian juga, sebuah sistem seharusnya tidak mengharuskan pengguna untuk mengetikkan informasi yang dapat dipilih dari daftar; memilih mengurangi kesalahan dan mempercepat entri.

Jenis Box	Kapan digunakan	Catatan
Check Box Menyajikan daftar pilihan lengkap, masing-masing dengan kotak persegi di depan	Ketika beberapa item dapat dipilih dari daftar item	• Kotak centang tidak saling eksklusif. Jangan gunakan negatif untuk label kotak. • Label kotak centang harus ditempatkan dalam beberapa urutan logis, seperti yang ditentukan oleh proses bisnis, atau gagal, menurut abjad atau paling umum digunakan terlebih dahulu. • Gunakan tidak lebih dari sepuluh kotak centang untuk serangkaian opsi tertentu. • Jika Anda membutuhkan lebih banyak kotak, kelompokkan ke dalam subkategori.
Radio Button Menyajikan daftar lengkap pilihan yang saling eksklusif, masing-masing dengan lingkaran di depan	Ketika hanya satu item yang dapat dipilih dari satu set item yang saling eksklusif	• Gunakan tidak lebih dari enam tombol radio dalam satu daftar; jika Anda membutuhkan lebih banyak, gunakan kotak daftar drop-down. • Jika hanya ada dua opsi, satu kotak centang biasanya lebih disukai daripada dua tombol radio, kecuali opsi tersebut tidak jelas. • Hindari menempatkan tombol radio di dekat kotak centang untuk mencegah kebingungan di antara daftar pilihan yang berbeda.
On-Screen List box		

Menyajikan daftar pilihan dalam kotak	Jarang atau tidak pernah—hanya jika tidak ada cukup ruang untuk kotak centang atau tombol radio	• Jenis kotak ini hanya dapat mengizinkan satu item untuk dipilih (dalam hal ini adalah versi tombol radio yang jelek). • Jenis kotak ini juga dapat mengizinkan banyak item untuk dipilih (dalam hal ini adalah versi kotak centang yang jelek), tetapi pengguna sering gagal untuk menyadari bahwa mereka dapat memilih beberapa item. • Jenis kotak ini memungkinkan daftar item untuk digulir, sehingga mengurangi jumlah ruang layar yang dibutuhkan.
Drop-Down list box		
Menampilkan item yang dipilih dalam kotak online yang terbuka untuk mengungkapkan daftar pilihan	Ketika tidak ada cukup ruang untuk menampilkan semua pilihan	• Kotak jenis ini berfungsi seperti tombol radio tetapi lebih ringkas. • Jenis kotak ini menyembunyikan pilihan dari pengguna hingga dibuka, yang dapat mengurangi kemudahan penggunaan; sebaliknya, karena melindungi pengguna pemula dari pilihan yang jarang digunakan, ini dapat meningkatkan kemudahan penggunaan. • Kotak jenis ini menyederhanakan desain jika jumlah pilihannya tidak jelas, karena hanya membutuhkan satu baris saat ditutup.
Combo box		
Jenis kotak daftar tarik-turun khusus yang memungkinkan pengguna mengetik serta menggulir daftar	Pintasan untuk pengguna berpengalaman	• Jenis kotak ini bertindak seperti daftar drop-down tetapi lebih cepat untuk pengguna berpengalaman ketika daftar item panjang.
Slider		
Skala grafis dengan penunjuk geser untuk memilih angka	Memasukkan perkiraan nilai numerik dari skala kontinu yang besar	• Slider mempersulit pengguna untuk memilih nomor yang tepat. • Beberapa penggeser juga menyertakan kotak nomor untuk memungkinkan pengguna memasukkan nomor tertentu.

Gambar 10-12 Jenis Kotak Pilihan

Dalam banyak kasus, beberapa bidang memiliki nilai yang sering berulang. Nilai yang sering ini harus digunakan sebagai nilai default untuk bidang sehingga pengguna dapat dengan mudah menerima nilai dan tidak perlu mengetik ulang berulang kali. Contoh nilai default adalah tanggal saat ini, kode area yang dipegang oleh mayoritas pelanggan perusahaan, dan alamat penagihan, yang didasarkan pada tempat tinggal pelanggan. Sebagian besar sistem mengizinkan perubahan ke nilai default untuk menangani pengecualian entri data saat terjadi.

Jenis Input

Setiap item data yang harus dimasukkan ditautkan ke bidang pada formulir tempat nilainya diketik. Setiap bidang juga memiliki label bidang, yaitu teks di samping, di atas, atau di bawah bidang yang memberi tahu pengguna jenis informasi apa yang termasuk dalam bidang tersebut. Seringkali label bidang mirip dengan nama elemen data, tetapi tidak harus memiliki kata yang identik. Dalam beberapa kasus, bidang menampilkan templat di atas kotak entri untuk menunjukkan kepada pengguna dengan tepat bagaimana data harus diketik. Ada banyak jenis input yang berbeda, dengan cara yang sama seperti ada banyak jenis field yang berbeda (lihat Gambar 10-12).

Teks. Seperti namanya, kotak teks digunakan untuk memasukkan teks. Kotak teks dapat didefinisikan memiliki panjang tetap atau dapat digulir dan dapat menerima jumlah teks yang hampir tidak terbatas. Dalam kedua kasus, kotak dapat berisi satu atau beberapa baris informasi tekstual. Kami tidak pernah menggunakan kotak teks jika kami dapat menggunakan kotak pilihan.

Kotak teks harus memiliki label bidang yang ditempatkan di sebelah kiri area entri, ukurannya jelas dibatasi oleh sebuah kotak (atau satu set garis bawah di antarmuka non-GUI). Jika ada beberapa kotak teks, label bidangnya dan tepi kiri kotak entrinya harus disejajarkan. Kotak teks harus mengizinkan fungsi GUI standar, seperti potong, salin, dan tempel.

Angka. Kotak angka digunakan untuk memasukkan angka. Beberapa perangkat lunak dapat secara otomatis memformat angka saat dimasukkan, sehingga 3452478 menjadi Rp. 34.524,78. Tanggal adalah bentuk angka khusus yang terkadang memiliki jenis kotak angka sendiri. Jangan pernah menggunakan kotak nomor jika Anda dapat menggunakan kotak pilihan.

Kotak Seleksi. Kotak pilihan memungkinkan pengguna untuk memilih nilai dari daftar yang telah ditentukan sebelumnya. Item dalam daftar harus diatur dalam beberapa urutan yang berarti, seperti abjad untuk daftar panjang atau dalam urutan yang paling sering digunakan. Nilai pemilihan default harus dipilih dengan hati-hati. Kotak pilihan dapat diinisialisasi sebagai tidak dipilih. Namun, lebih baik memulai dengan item yang paling sering digunakan yang sudah dipilih.

Validasi Input

Semua data yang dimasukkan ke dalam sistem perlu divalidasi untuk memastikan keakuratannya. Validasi input (juga disebut pemeriksaan edit) dapat dilakukan dalam berbagai bentuk. Idealnya, sistem komputer tidak boleh menerima data yang gagal dalam pemeriksaan validasi penting untuk mencegah informasi yang tidak valid memasuki sistem. Namun, ini bisa sangat sulit, dan data yang tidak valid sering kali melewati operator entri data dan pengguna yang memberikan informasi. Terserah sistem untuk mengidentifikasi data yang tidak valid dan membuat perubahan atau memberi tahu seseorang yang dapat menyelesaikan masalah informasi.

Ada enam jenis pemeriksaan validasi: pemeriksaan kelengkapan, pemeriksaan format, pemeriksaan rentang, pemeriksaan digit, pemeriksaan konsistensi, dan pemeriksaan database (lihat Gambar 10-13). Setiap sistem harus menggunakan setidaknya satu pemeriksaan validasi pada semua data yang dimasukkan dan, idealnya, melakukan semua pemeriksaan yang sesuai jika memungkinkan.

Jenis Validasi	Kapan digunakan	Catatan
Completeness check		
Memastikan semua data yang diperlukan telah dimasukkan	Ketika beberapa bidang harus dimasukkan sebelum formulir dapat diproses	Jika informasi yang diperlukan tidak ada, formulir dikembalikan ke pengguna tanpa diproses.
Format check		
Pastikan data memiliki jenis yang benar (misalnya, numerik) dan dalam format yang benar (misalnya, bulan, hari, tahun)	Ketika bidang numerik atau berisi data berkode	Idealnya, bidang numerik tidak boleh mengizinkan pengguna untuk mengetik data teks, tetapi jika ini tidak memungkinkan, data yang dimasukkan harus diperiksa untuk memastikannya numerik. Beberapa bidang menggunakan kode atau format khusus (mis., Plat nomor dengan tiga huruf dan tiga angka) yang harus diperiksa.
Range check		
Pastikan data numerik berada dalam nilai minimum dan maksimum yang benar	Dengan semua data numerik, jika memungkinkan	Pemeriksaan rentang hanya mengizinkan angka di antara nilai yang benar. Sistem seperti itu juga dapat digunakan untuk menyaring data untuk "kewajaran"—misalnya, menolak tanggal lahir sebelum tahun 1880 karena orang tidak hidup lebih dari 100 tahun (kemungkinan besar, tahun 1980 dimaksudkan).
Check digit check		
Periksa digit ditambahkan ke kode numerik	Ketika kode numerik digunakan	Digit cek adalah angka yang ditambahkan ke kode sebagai cara untuk memungkinkan sistem memvalidasi kebenaran dengan cepat. Misalnya, nomor Jaminan Sosial AS dan nomor Asuransi Sosial Kanada hanya menetapkan delapan dari sembilan digit dalam nomor tersebut. Angka kesembilan—digit cek—dihitung menggunakan rumus matematika dari delapan angka pertama. Ketika nomor identifikasi diketik ke dalam sistem komputer, sistem menggunakan rumus dan membandingkan hasilnya dengan digit cek. Jika angka tidak cocok, maka telah terjadi kesalahan.
Consistency checks		

Pastikan kombinasi data valid	Ketika data terkait	Bidang data sering terkait. Misalnya, tahun kelahiran seseorang harus mendahului tahun di mana dia menikah. Meskipun tidak mungkin bagi sistem untuk mengetahui data mana yang salah, sistem dapat melaporkan kesalahan tersebut kepada pengguna untuk diperbaiki.
Database checks		
Bandingkan data dengan database (atau file) untuk memastikan mereka benar	Ketika data tersedia untuk diperiksa	Data dibandingkan dengan informasi dalam database (atau file) untuk memastikan kebenarannya. Misalnya, sebelum nomor identifikasi diterima, database ditanyai untuk memastikan bahwa nomor tersebut valid. Karena pemeriksaan basis data lebih mahal daripada jenis pemeriksaan lainnya (memerlukan sistem untuk melakukan lebih banyak pekerjaan), sebagian besar sistem melakukan pemeriksaan lain terlebih dahulu dan melakukan pemeriksaan basis data hanya setelah data melewati pemeriksaan sebelumnya.

Gambar 10-13 Jenis Validasi Input

10.7 OUTPUT DESAIN

Output adalah apa yang dihasilkan sistem, baik di layar, di atas kertas, atau di media lain, seperti Web. Output mungkin merupakan bagian yang paling terlihat dari sistem apapun karena alasan utama menggunakan sistem informasi adalah untuk mengakses informasi yang dihasilkannya.

Prinsip dasar

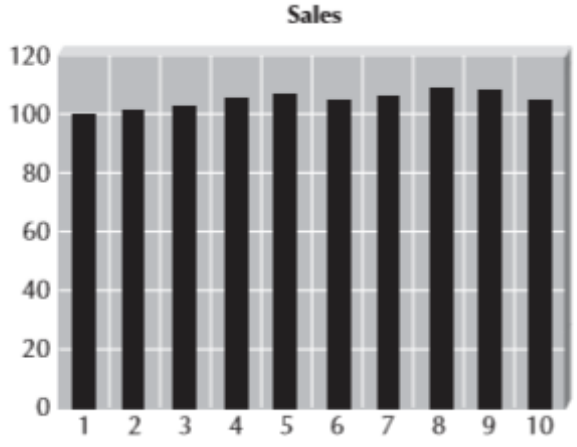
Tujuan dari mekanisme Output adalah untuk menyajikan informasi kepada pengguna sehingga mereka dapat secara akurat memahaminya dengan sedikit usaha. Prinsip-prinsip dasar untuk desain output mencerminkan bagaimana output digunakan dan cara untuk membuatnya lebih mudah bagi pengguna untuk memahaminya.

Memahami Penggunaan Laporan. Prinsip pertama dalam mendesain laporan adalah memahami cara penggunaannya. Laporan dapat digunakan untuk berbagai tujuan. Dalam beberapa kasus—tetapi tidak terlalu sering—laporan dibaca dari depan ke belakang karena semua informasi diperlukan. Dalam kebanyakan kasus, laporan digunakan untuk mengidentifikasi item tertentu atau digunakan sebagai referensi untuk menemukan informasi, sehingga urutan item yang diurutkan pada laporan atau dikelompokkan dalam kategori sangat penting. Ini sangat penting untuk desain laporan elektronik atau berbasis Web. Laporan web yang dimaksudkan untuk dibaca dari awal hingga akhir harus disajikan dalam satu halaman panjang yang dapat digulir, sedangkan laporan yang digunakan terutama untuk menemukan informasi spesifik harus dipecah menjadi beberapa halaman, masing-masing dengan tautan terpisah. Nomor halaman dan tanggal penyusunan laporan juga penting untuk laporan referensi.

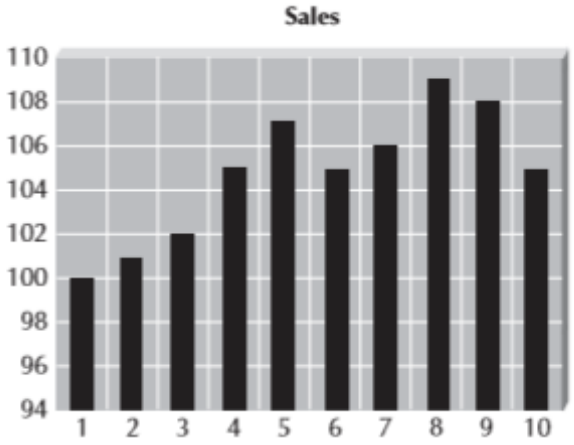
Frekuensi laporan juga dapat memainkan peran penting dalam desain dan distribusinya. Laporan real-time memberikan data yang akurat hingga detik atau menit saat mereka diproduksi (mis., kutipan pasar saham). Laporan batch adalah laporan yang melaporkan informasi historis yang mungkin berumur beberapa bulan, hari, atau jam, dan sering kali memberikan informasi tambahan di luar informasi yang dilaporkan (misalnya, total, ringkasan, rata-rata historis).

Tidak ada keuntungan yang melekat pada laporan real-time dibandingkan laporan batch. Satu-satunya keuntungan terletak pada nilai waktu informasi. Jika informasi dalam laporan bersifat time critical (misalnya, harga saham, informasi kontrol lalu lintas udara), maka laporan real-time memiliki nilai. Ini sangat penting karena laporan waktu nyata seringkali mahal untuk diproduksi; kecuali mereka menawarkan nilai bisnis yang jelas, mereka mungkin tidak sepadan dengan biaya tambahannya.

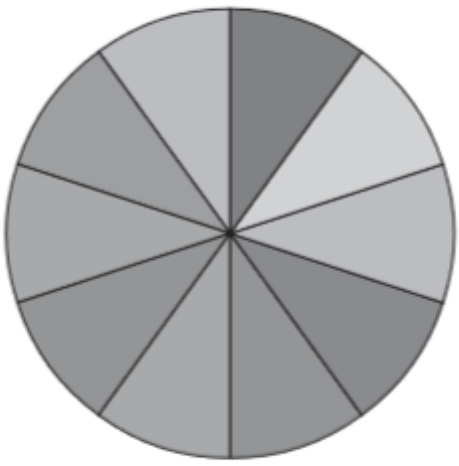
Kelola Beban Informasi. Kebanyakan manajer mendapatkan terlalu banyak informasi, tidak terlalu sedikit (yaitu, beban informasi yang harus ditangani manajer terlalu besar). Tujuan dari laporan yang dirancang dengan baik adalah untuk menyediakan semua informasi yang diperlukan untuk mendukung tugas yang dirancang. Ini tidak berarti bahwa laporan harus menyediakan semua informasi yang tersedia tentang subjek—hanya apa yang diputuskan pengguna untuk melakukan pekerjaan mereka. Dalam beberapa kasus, hal ini dapat menghasilkan beberapa laporan berbeda tentang topik yang sama untuk pengguna yang sama karena digunakan dengan cara yang berbeda. Ini bukan desain yang buruk.

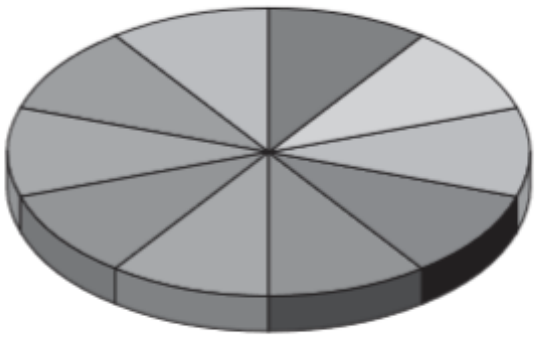


(a) Unbiased graph with scale starting at 0



(b) Biased graph with scale starting at 94


(c) Unbiased graph in 2D


(d) Biased graph in 3D

Gambar 10-14 Bias dalam Grafik

Untuk pengguna di negara-negara kebarat-baratan, informasi terpenting harus selalu disajikan terlebih dahulu di sudut kiri atas layar atau laporan kertas. Informasi harus disediakan dalam format yang dapat digunakan tanpa modifikasi. Pengguna tidak perlu mengurutkan ulang informasi laporan; sebaliknya informasi penting harus disorot sehingga pengguna dapat menemukannya dengan lebih mudah di tengah banyak data, atau melakukan perhitungan matematis tambahan.

Minimalkan Bias. Tidak ada analis yang merancang laporan yang bias. Masalah dengan bias adalah bahwa hal itu bisa sangat halus; analis dapat memperkenalkannya secara tidak sengaja. Bias dapat diperkenalkan dengan cara daftar data diurutkan karena entri yang muncul pertama kali dalam daftar dapat menerima lebih banyak perhatian daripada entri yang muncul belakangan dalam daftar. Data sering diurutkan dalam urutan abjad, membuat entri yang dimulai dengan huruf A lebih menonjol. Data dapat diurutkan dalam urutan kronologis (atau urutan kronologis terbalik), dengan lebih menekankan pada entri yang lebih lama (atau terbaru). Data dapat diurutkan berdasarkan nilai numerik, lebih menekankan pada nilai yang lebih tinggi atau lebih rendah. Misalnya, pertimbangkan laporan penjualan bulanan menurut negara bagian. Haruskah laporan dicantumkan dalam urutan abjad menurut nama negara bagian, dalam urutan menurun menurut jumlah yang terjual, atau dalam urutan lain (misalnya, wilayah geografis)? Tidak ada jawaban yang mudah untuk ini, kecuali untuk mengatakan bahwa urutan penyajian harus sesuai dengan cara penggunaan informasi.

Tampilan dan laporan grafis dapat menghadirkan masalah desain yang sangat menantang. Skala pada sumbu dalam grafik sangat rentan terhadap bias. Untuk sebagian besar jenis grafik, skala harus selalu dimulai dari nol; jika tidak, perbandingan di antara nilai-nilai dapat menyesatkan. Misalnya, apakah penjualan meningkat sangat banyak sejak tahun 1 (lihat Gambar 10-14a dan b)? Angka-angka di kedua grafik itu sama, tetapi gambar visual yang disajikan keduanya sangat berbeda. Pandangan sekilas pada Gambar 10-14a hanya akan menunjukkan perubahan kecil, sedangkan pandangan sekilas pada Gambar 10-14b mungkin menunjukkan bahwa ada beberapa peningkatan yang signifikan. Bahkan, penjualan telah meningkat sebesar 15 persen selama lima tahun, atau 3 persen per tahun. Gambar 10-14a menyajikan gambar yang paling akurat; Gambar 10-14b bias karena skala dimulai sangat dekat dengan nilai terendah dalam grafik dan menyesatkan mata untuk menyimpulkan bahwa telah terjadi perubahan besar. Anda juga harus berhati-hati terhadap apa yang disebut efek 3D. Misalnya, diagram lingkaran pada Gambar 10-14c dan d mewakili data yang sama; sebenarnya data itu sendiri adalah konstan. Namun, karena diagram lingkaran “3D”, irisan di dekat bagian depan terlihat lebih besar.

Jenis Output

Ada banyak jenis laporan yang berbeda, seperti laporan detail, laporan ringkasan, laporan pengecualian, dokumen turnaround, dan grafik (lihat Gambar 10-15). Mengklasifikasikan laporan merupakan tantangan karena banyak laporan memiliki karakteristik dari beberapa jenis yang berbeda. Misalnya, beberapa laporan detail juga menghasilkan total ringkasan, menjadikannya laporan ringkasan.

Media

Berbagai jenis media digunakan untuk menghasilkan laporan. Saat ini, sebagian besar organisasi telah beralih ke "pencetakan" laporan secara elektronik. Salah satu format yang populer adalah pdf Adobe. “Laporan” ini disimpan dalam format elektronik di server file atau server Web sehingga pengguna dapat

Penjualan dengan mudah mengaksesnya. Seringkali laporan tersedia dalam format yang lebih dirancang sebelumnya daripada rekan-rekan mereka yang berbasis kertas karena biaya produksi dan penyimpanan format yang berbeda minimal. Laporan elektronik juga dapat diproduksi sesuai permintaan sesuai kebutuhan, dan memungkinkan pengguna untuk lebih mudah mencari kata-kata tertentu. Selanjutnya, laporan elektronik dapat menyediakan sarana pendukung laporan ad hoc, di mana pengguna menyesuaikan isi laporan pada saat laporan dibuat. Beberapa pengguna masih mencetak laporan elektronik pada printer mereka sendiri, tetapi pengurangan biaya pengiriman elektronik jarak jauh dan kemudahan yang memungkinkan lebih banyak pengguna untuk mengakses laporan daripada ketika mereka hanya dalam bentuk kertas biasanya diimbangi dengan biaya pencetakan lokal.

Jenis Laporan/Report	**Kapan digunakan**	**Catatan**
Detail Report		
Daftar informasi rinci tentang semua item yang diminta	Ketika pengguna membutuhkan informasi lengkap tentang item	• Laporan ini biasanya dibuat hanya sebagai tanggapan atas pertanyaan tentang item yang cocok dengan beberapa kriteria. • Laporan ini biasanya dibaca sampul ke sampul untuk membantu pemahaman satu atau lebih item secara mendalam.
Summary Report		
Mencantumkan informasi ringkasan tentang semua item	Ketika pengguna membutuhkan informasi singkat tentang banyak item	• Laporan ini biasanya dibuat hanya sebagai tanggapan atas kueri tentang item yang cocok dengan beberapa kriteria, tetapi dapat berupa database yang lengkap. • Laporan ini biasanya dibaca untuk tujuan membandingkan beberapa item satu sama lain. • Urutan di mana item diurutkan adalah penting.
Turnaround document		
Keluaran yang “berbalik” dan menjadi masukan	Ketika pengguna (seringkali pelanggan) perlu mengembalikan output untuk diproses	• Dokumen turnaround adalah jenis laporan khusus yang merupakan output dan input. • Misalnya, sebagian besar tagihan yang dikirim ke konsumen (misalnya tagihan kartu kredit) memberikan informasi tentang jumlah total yang terutang dan juga berisi formulir yang diisi dan dikembalikan oleh konsumen dengan pembayaran.
Graphs		
Bagan yang digunakan sebagai tambahan dan	Ketika pengguna perlu membandingkan	• Grafik yang dibuat dengan baik membantu pengguna membandingkan dua atau lebih item atau memahami

sebagai pengganti tabel angka	data di antara beberapa item	bagaimana satu item telah berubah dari waktu ke waktu. • Grafik buruk dalam membantu pengguna mengenali nilai numerik yang tepat dan harus diganti dengan atau digabungkan dengan tabel ketika presisi penting. • Bagan batang cenderung lebih baik daripada tabel angka atau jenis bagan lain dalam hal membandingkan nilai antar item (tetapi hindari bagan tiga dimensi yang membuat perbandingan menjadi sulit). • Bagan garis memudahkan untuk membandingkan nilai dari waktu ke waktu, sedangkan bagan sebar memudahkan untuk menemukan klaster atau data yang tidak biasa. • Diagram lingkaran menunjukkan proporsi atau bagian relatif dari keseluruhan.

Gambar 10-15 Jenis Laporan

10.8 KOMPUTASI SELULER DAN DESAIN ANTARMUKA PENGGUNA

Dari perspektif desain antarmuka pengguna, menggunakan perangkat seluler itu mengasyikkan sekaligus menantang. Jelas, dengan smartphone saat ini, seperti Droid™ atau iPhone™, ada banyak kemungkinan. Namun, hanya karena ponsel ini memiliki kemampuan untuk menjelajahi Web tidak berarti bahwa antarmuka Web yang sederhana adalah jawabannya. Perangkat ini memiliki ruang layar terbatas dan memiliki kemampuan, seperti layar sentuh dan umpan balik haptic (seperti getaran atau pulsa), yang tidak dimiliki komputer biasa. Akibatnya, Anda benar-benar perlu fokus pada perancangan antarmuka untuk perangkat dan tidak hanya memindahkan antarmuka Web ke perangkat tersebut. Selanjutnya, Anda perlu menyadari bahwa tablet, seperti iPad™, bukanlah smartphone besar; itu dalam kategorinya sendiri dengan tantangan dan kemampuannya sendiri. Akibatnya, Anda benar-benar perlu mendesain antarmuka untuk perangkat seluler dari awal. Di bagian ini, kami membahas beberapa tantangan dan memberikan beberapa panduan untuk mengembangkan antarmuka seluler yang efektif. Namun, sebelum kita mulai, Anda harus menyadari bahwa semua materi yang dijelaskan sebelumnya masih berlaku. Hanya saja ketika Anda berhadapan dengan perangkat tersebut, masalah tambahan harus diperhatikan.

Tidwell mengidentifikasi enam tantangan yang harus dihadapi oleh perancang antarmuka pengguna seluler. Layar ponsel kecil. Tidak banyak "real estat" yang tersedia untuk digunakan. Tidak hanya layarnya kecil, tetapi mereka datang dalam berbagai ukuran. Apa yang berfungsi di satu layar mungkin tidak berfungsi di layar lain. Beberapa layar memiliki kemampuan haptic: Mereka merespons sentuhan dan orientasi, dan dalam beberapa kasus, mereka bergetar. Jelas, kemampuan ini tidak tersedia di semua perangkat seluler. Namun, mereka memberikan kemungkinan menarik untuk desain antarmuka pengguna. Keypad fisik virtual dan aktual berukuran kecil. Akibatnya, terlalu banyak mengetik dapat menjadi tantangan bagi pengguna untuk memasukkan informasi yang benar. Orang-orang menggunakan perangkat seluler mereka, terutama ponsel mereka, di semua jenis lingkungan. Mereka menggunakannya di tempat-tempat gelap (seperti ruang kelas dengan penerangan

yang buruk). Mereka menggunakannya di bawah sinar matahari yang cerah. Mereka menggunakannya di tempat yang tenang (seperti perpustakaan atau bioskop) dan mereka menggunakannya di tempat yang bising (seperti di pertandingan sepak bola). Perangkat ini hanya digunakan di mana-mana hari ini. Karena perangkat ini digunakan di mana-mana, pengguna dapat dengan mudah dialihkan dari perangkat. Misalnya, apakah Anda pernah mengirim pesan teks kepada seseorang saat Anda tidak seharusnya menggunakan ponsel, seperti di kelas? Atau, bagaimana dengan kencan? Dengan kata lain, pengguna biasanya multitasking saat menggunakan ponsel mereka. Mereka tidak ingin menghabiskan banyak energi untuk mencoba menavigasi situs atau aplikasi seluler Anda. Akibatnya, tiga prinsip desain Krug yang dijelaskan sebelumnya sangat penting, terutama yang pertama: Jangan membuat saya berpikir!

Berdasarkan tantangan ini, Tidwell memberikan serangkaian saran yang harus Anda ikuti dalam merancang antarmuka pengguna untuk perangkat ini. Pertama, mengingat konteks seluler, Anda benar-benar perlu fokus pada apa yang dibutuhkan pengguna dan bukan apa yang mungkin diinginkan pengguna. Dengan kata lain, Anda benar-benar harus kembali ke proses bisnis dan pemodelan fungsional (Bab 4). Dalam hal ini, hanya fokus pada tugas yang perlu dilakukan pengguna saat mereka berada dalam konteks seluler. Ini adalah contoh yang baik dari persyaratan nonfungsional (komputasi seluler) yang memengaruhi kemungkinan persyaratan fungsional.

Gambar 10-16 Contoh Linearisasi, Pusat Studi Bisnis Wiley, digunakan dengan izin.

Kedua, jika Anda mem-porting aplikasi atau situs web ke perangkat seluler, hapus semua "bulu halus" dari situs: Hapus situs hingga bagian dasarnya. Jika pengguna memerlukan akses ke situs lengkap, pastikan untuk memberikan tautan ke situs tersebut di lokasi yang jelas. Atau, Anda dapat memberikan versi seluler lengkap dari aplikasi atau situs web kepada pengguna. Jelas, desain antarmuka pengguna akan berbeda, tetapi fungsinya harus sama.

Ketiga, bila memungkinkan, manfaatkan kemampuan unik yang ada pada perangkat ini. Beberapa perangkat memiliki GPS bawaan. Tergantung pada aplikasi Anda, mengetahui di mana pengguna berada dapat mengubah hasilnya. Dalam kasus lain, perangkat memiliki akselerometer yang memungkinkan aplikasi untuk "mengetahui" orientasi perangkat. Banyak dari perangkat ini memiliki kemampuan pengenalan suara, kamera yang dapat digunakan untuk pemindaian, layar sentuh yang memungkinkan penggunaan gerakan canggih, dan umpan balik haptic, seperti benturan dan getaran. Semua kemampuan ini terbukti berguna dalam mengembangkan berbagai aplikasi seluler.

Keempat, ketika mempertimbangkan telepon, Anda cenderung memiliki lebar yang terbatas untuk bekerja. Akibatnya, Anda harus mencoba membuat linierisasi konten aplikasi (lihat Gambar 10-16).

Kelima, optimalkan aplikasi seluler Anda untuk pengguna. Ini termasuk meminimalkan berapa kali perangkat harus berinteraksi dengan server untuk mengunduh atau mengunggah informasi dengan server. Tidak semua orang memiliki akses ke 3G, hanya jaringan 4G sejati. Dalam banyak kasus, pengunggahan dan pengunduhan masih sangat lambat. Pengoptimalan juga mencakup interaksi pengguna dengan perangkat. Alih-alih menggunakan banyak mengetik, menggulir, dan mengetuk layar sentuh, pertimbangkan untuk menggunakan kemampuan pengenalan suara. Jauh lebih mudah untuk berbicara perlahan ke smartphone daripada harus banyak mengetik di keyboard virtual atau fisik.

Tidwell juga menyediakan serangkaian pola yang dapat digunakan kembali yang telah disesuaikan untuk perangkat seluler. Ini termasuk hal-hal seperti tumpukan vertikal, strip film, dan navigasi bawah.

Selain saran umum yang diberikan Tidwell, seluruh area interaksi harus dirancang. Dengan antarmuka berbasis GUI tradisional, interaksi cenderung terbatas pada mengetik di keyboard; menggunakan mouse untuk mengklik, menggulir, atau memperbesar antarmuka; atau menggunakan kombinasi keyboard dan mouse untuk memutar sebagian konten di antarmuka. Secara keseluruhan, ini adalah serangkaian interaksi yang cukup terbatas. Namun, dengan perangkat seluler baru yang memiliki pengenalan suara, pembuatan suara, layar sentuh, umpan balik haptic (melalui getaran), akselerometer yang memungkinkan perangkat mengetahui orientasinya, dan kamera yang dapat digunakan untuk memindai input, sejumlah opsi untuk merancang navigasi dan bagian input dan output dari antarmuka pengguna telah meningkat secara substansial. Dari sudut pandang kami, kami baru saja mendeteksi puncak gunung es pepatah kemungkinan dengan perangkat ini. Namun, pendekatan prototyping umum yang disarankan sebelumnya dalam bab ini berhasil. Hanya jumlah opsi yang perlu dipertimbangkan telah meningkat secara substansial.

Ketika datang ke bagian navigasi antarmuka pengguna, opsi tambahan utama adalah penggunaan layar sentuh. Faktanya, ada banyak kosakata tentang cara pengguna berinteraksi dengan layar sentuh. Saat ini, perancang perlu mempertimbangkan untuk mengetuk, mencubit, menyebarkan, menjentikkan, menggulir (satu jari vs. dua jari), dan menyeret untuk beberapa nama. Sebagai contoh:

- Mengetuk dapat digunakan untuk membuka atau mengaktifkan aplikasi, untuk memilih objek di antarmuka, atau untuk menghentikan tindakan, seperti menggulir.
- Mencubit digunakan untuk mengecilkan atau memperkecil.
- Spreading digunakan untuk memperbesar atau memperbesar.
- Menjentikkan dapat digunakan untuk memindahkan objek atau untuk berinteraksi dengan penggeser untuk menggulir.
- Menggulir dapat dilakukan dengan menggunakan satu jari pada bilah gulir atau dengan menggunakan dua jari di mana saja pada antarmuka.
- Pindahkan objek di layar dengan meletakkan jari pada objek dan menyeretnya ke lokasi lain. Ini mirip dengan menggunakan mouse untuk menyeret objek ke lokasi lain.

Perangkat seluler yang berbeda dapat menerapkan masing-masing sedikit berbeda. Jelas, jumlah pilihan untuk mendukung navigasi telah sangat berkembang.

Untuk bagian input antarmuka pengguna, opsi tambahan utama yang perlu dipertimbangkan adalah penggunaan kamera sebagai perangkat untuk memindai item; mikrofon sebagai perangkat input ucapan; dan akselerometer untuk mendeteksi akselerasi, orientasi, dan getaran. Misalnya, Amazon Mobile™ dan Red Laser Barcode & QR Scanner™ dari eBay Mobile menyediakan sarana untuk memindai barcode produk dengan mudah dan dengan cepat mencari produk secara online untuk menyediakan semua jenis informasi tentang produk. Google Maps™ mendukung antarmuka pengenalan suara untuk menemukan lokasi. Selain itu, ada banyak aplikasi yang menggunakan akselerometer untuk mendeteksi akselerasi dan getaran (misalnya, Accelerometer Data Pro™ dari Wavefront Labs) dan untuk menghitung orientasi (misalnya, NH untuk Easy Spirit Level™ dari Mobikats). Selain itu, banyak game yang tersedia di perangkat seluler menggunakan akselerometer sebagai perangkat input.

Opsi tambahan utama yang perlu dipertimbangkan untuk bagian keluaran antarmuka pengguna termasuk pembuatan suara dan umpan balik haptic. Misalnya, Google Maps™ dan Waze™ mendukung kemampuan menghasilkan suara yang memberikan petunjuk arah mengemudi. Selain itu, hampir semua smartphone mendukung opsi getar saat telepon berdering dan saat peringatan atau pesan teks tiba.

10.9 MEDIA SOSIAL DAN DESAIN ANTARMUKA PENGGUNA

Mengingat dampak yang dimiliki Facebook™ dan Twitter™, misalnya, pemberontakan Arab di dunia saat ini, pengembangan aplikasi untuk media sosial jelas menjadi yang terdepan. Dalam banyak hal, komputasi seluler dan media sosial telah tumbuh bersama. Seperti halnya komputasi seluler, setiap platform media sosial memiliki kemampuan dan tantangannya sendiri. Platform media sosial berkisar dari situs yang memungkinkan Anda mengunggah materi ke sana, seperti Flickr™ dan YouTube™, hingga situs yang mendukung keberadaan virtual di metaverse, seperti Second Life™. Selama karir Anda, Anda mungkin perlu mengembangkan aplikasi untuk platform media sosial tertentu, seperti Facebook™ atau Twitter™, atau mungkin mengembangkan situs media sosial Anda sendiri.

Saat mengembangkan kehadiran media sosial Anda sendiri, Anda harus memahami siapa audiens target Anda. Apakah audiens adalah karyawan perusahaan Anda, atau audiens di luar perusahaan? Di bagian ini, kami hanya fokus pada audiens eksternal. Setelah Anda mengetahui siapa audiensnya, Anda perlu mengetahui apa yang mereka katakan tentang perusahaan tersebut. Dalam banyak hal, media sosial tidak lebih dari saluran lain untuk memasarkan produk dan kemampuan perusahaan. Sebelum Anda dapat menyebarkan kehadiran media sosial, Anda benar-benar perlu memahami apa kebutuhan (keinginan) pengguna. Dengan kata lain, kembali ke penentuan kebutuhan. Dalam hal ini, masalahnya

adalah bahwa pengguna "di luar sana" di suatu tempat, sehingga pendekatan khas untuk mengumpulkan persyaratan, seperti wawancara dan observasi, tidak berfungsi. Sebaliknya, Anda perlu menelusuri Web untuk membasmi kebutuhan Anda. Beberapa tempat yang lebih berguna untuk dilihat adalah blog atau outlet media sosial lainnya yang membahas masalah yang akan menarik bagi perusahaan Anda. Ketika semuanya gagal, Anda selalu dapat menggunakan mesin pencari seperti Google™. Apapun, Anda jelas harus memahami persyaratan fungsional sebelum Anda dapat merancang kehadiran media sosial Anda.

Setelah Anda memahami persyaratan fungsional Anda, Anda perlu menentukan jenis kehadiran media sosial apa yang diperlukan untuk memenuhi persyaratan secara efektif. Setiap platform media sosial memiliki ceruknya sendiri. Akibatnya, Anda mungkin perlu menerapkan banyak aplikasi berbeda di berbagai platform untuk secara efektif memenuhi persyaratan kehadiran media sosial perusahaan. Juga, Anda harus melihat situs media sosial Anda sebagai sarana bagi perusahaan Anda untuk membangun dan mempertahankan citra atau merek yang positif. Oleh karena itu, situs media sosial harus memuat materi yang ingin dikonsumsi oleh calon pelanggan Anda. Anda harus ingat bahwa tujuan yang mendasari pemasaran adalah untuk "memproduksi" keinginan dan kemudian "mengubah" keinginan menjadi kebutuhan. Mengingat bahwa situs media sosial Anda secara efektif merupakan saluran pemasaran lain, situs Anda harus dapat menarik pelanggan baru dan membuat pelanggan saat ini kembali secara teratur. Di bagian ini, kami memberikan beberapa panduan umum untuk mengembangkan situs media sosial Anda sendiri sehingga pelanggan baru berkunjung dan pelanggan lama kembali.

Pertama, Anda benar-benar perlu memposting ke situs Anda secara teratur. Jika konten situs menjadi stagnan, tidak ada yang mau mengunjungi. Konten situs harus berisi campuran media: video, podcast, klip suara, dan sebagainya. Materi situs harus mencakup campuran materi yang digerakkan oleh perusahaan, materi dari pelanggan, dan tautan ke konten relevan yang terletak di situs lain. Juga, pastikan untuk menyertakan cara bagi pengunjung untuk bergabung dalam "percakapan" dengan perusahaan, seperti komentar Facebook™ atau Tweet Twitter™.

Kedua, pastikan Anda memahami perbedaan antara pendekatan push dan pull. Jika pengguna harus datang kepada Anda untuk mencari tahu sesuatu, maka Anda menggunakan pendekatan berbasis tarik. Di sisi lain, jika Anda memberikan informasi kepada pengguna, maka Anda menggunakan pendekatan berbasis push. Ketika datang ke media sosial, Anda benar-benar perlu menggunakan kombinasi pendekatan. Misalnya, di Facebook™ jika seseorang memposting di dinding Anda atau mengirimi Anda permintaan, Facebook™ akan mengirimi Anda pesan email untuk mencoba membujuk Anda kembali ke situs Facebook™. Tindakan memposting ke situs Anda adalah tindakan berbasis tarikan, dan pesan email yang dikirimkan kepada Anda adalah tindakan berbasis push. Singkatnya, Anda ingin lebih fokus pada pendekatan berbasis push. Anda ingin konten Anda sampai ke pelanggan Anda seefektif mungkin. Anda tidak ingin mereka harus datang mencari Anda. Dorong mereka untuk ikut serta agar pemberitahuan pembaruan datang kepada mereka dalam bentuk yang mereka sukai. Beberapa mungkin lebih suka pemberitahuan email, dan yang lain mungkin lebih suka Anda memposting ke akun Facebook™ atau Twitter™ mereka. Juga, pastikan untuk menyertakan tautan ke situs media sosial Anda di halaman beranda. Tapi pastikan untuk tidak membanjiri pelanggan. Tidak setiap pelanggan ingin mengetahui setiap informasi menarik tentang perusahaan. Hanya memberikan pelanggan apa yang pelanggan inginkan. Ingat, prinsip pertama Krug: Jangan membuatku berpikir! Akibat wajar dari prinsip media sosial ini adalah: Jangan buat saya bekerja! Permudah pelanggan untuk menemukan hanya apa yang mereka inginkan (atau mungkin apa yang kita ingin mereka inginkan).

Ketiga, pastikan bahwa halaman rumah Anda dan situs media sosial Anda semua disinkronkan bersama sehingga ketika satu diperbarui, situs lain "tahu" tentang pembaruan tersebut. Hal ini membuat pekerjaan Anda dalam memelihara situs yang berbeda jauh lebih mudah, dan memungkinkan pelanggan Anda untuk memiliki pengalaman yang konsisten di semua situs. Namun, jangan berlebihan. Jelas bahwa situs yang berbeda memiliki media yang berbeda dan, berpotensi, audiens yang berbeda. Anda tidak akan menggunakan Facebook™ dengan cara yang sama seperti Anda menggunakan Twitter™, YouTube™, atau blog. Pastikan untuk menyertakan tautan silang di antara situs yang berbeda. Ini memungkinkan pelanggan Anda untuk dengan mudah menavigasi melalui situs Anda yang berbeda.

Keempat, memungkinkan pelanggan untuk berbagi konten hebat yang telah Anda buat. Anda dapat menyertakan tombol yang memungkinkan mereka mengirim konten melalui email ke "teman" terdekat mereka atau pengikut lain di jejaring sosial mereka sendiri. Anda juga harus menyediakan sarana untuk mengumpulkan umpan balik dari pelanggan Anda mengenai konten Anda. Salah satu caranya adalah dengan menyertakan kemampuan pelanggan untuk membuat dan membagikan komentar terkait konten Anda. Cara lain adalah dengan menyediakan mekanisme voting atau “suka” untuk mendorong pelanggan agar terlibat dengan situs Anda.

Kelima, pastikan untuk merancang situs Anda sehingga tidak hanya pelanggan Anda yang dapat dengan mudah menemukan materi yang mereka cari, tetapi juga mesin pencari dapat menemukan materi tersebut. Mesin pencari setidaknya memiliki kemungkinan yang sama untuk mendatangkan pelanggan baru ke situs Anda seperti halnya pelanggan lain. Rancang situs sehingga begitu pelanggan tiba di situs Anda, dia akan tinggal di sana untuk sementara waktu. Salah satu cara yang dapat Anda lakukan adalah dengan menyediakan tautan ke materi "terkait" kepada pelanggan. Jika Anda memutuskan untuk menyertakan mekanisme pemungutan suara atau "suka", pastikan untuk memungkinkan pelanggan melihat materi "terbaik" atau, setidaknya, paling populer terlebih dahulu. Kemungkinan lain adalah membuat papan peringkat yang menampilkan materi yang paling banyak dibagikan. Anda perlu memanfaatkan informasi yang diperoleh dengan menerapkan pedoman keempat.

Keenam, salah satu hal yang lebih sulit untuk dicapai adalah membuat situs Anda menjadi tempat yang menurut pelanggan Anda milik mereka. Anda ingin pelanggan Anda merasa bahwa mereka adalah anggota dari sesuatu; Anda ingin mencoba membangun perasaan komunitas. Semakin mereka merasa bahwa mereka termasuk, semakin besar kemungkinan mereka akan merekomendasikan situs Anda kepada teman-teman mereka. Salah satu cara untuk mencapai ini adalah dengan mendorong karyawan, setidaknya karyawan yang "benar", untuk membuat situs "independen" mereka sendiri yang membahas topik yang menarik bagi pelanggan Anda. Ini akan memberikan perasaan yang lebih pribadi kepada perusahaan dan mungkin menarik pelanggan untuk tetap berada di situs lebih lama.

Terakhir, dalam banyak kasus, pelanggan Anda mengunjungi situs Anda menggunakan berbagai platform perangkat keras. Platform berkisar dari desktop ke notebook ke tablet ke smartphone. Akibatnya, semua materi yang terkait dengan desain antarmuka pengguna umum dan komputasi seluler dapat diterapkan. Karena Anda memiliki audiens global, Anda harus memastikan untuk mempertimbangkan isu-isu internasional dan budaya dalam desain Anda.

10.10 GAME, VISUALISASI INFORMASI MULTIDIMEN, DAN LINGKUNGAN YANG MENAKJUBKAN

Dengan munculnya game dan visualisasi informasi multidimensi yang digunakan dalam bisnis dan potensi penerapan teknologi imersif, seperti Google Glass™ dan Oculus Rift™, untuk memecahkan masalah bisnis menggunakan augmented reality dan virtual reality,

desain lapisan interaksi komputer manusia menjadi bahkan lebih penting dalam pengembangan sistem informasi. Dalam banyak hal, desain antarmuka pengguna untuk game, visualisasi informasi multidimensi, dan lingkungan imersif sangat mirip dengan merancang antarmuka pengguna untuk area aplikasi yang lebih tradisional. Namun, dalam hal lain, sangat berbeda.

Game, Gamification, dan Desain Antarmuka Pengguna

Game telah ada untuk waktu yang sangat, sangat lama. Mereka telah sangat sukses di banyak bidang yang berbeda karena mereka menyenangkan dan menarik. Saat menerapkan game ke situasi bisnis, ada dua pendekatan umum yang perlu dipertimbangkan: pengembangan game yang mendukung proses bisnis dan gamifikasi proses bisnis. Perkembangan game untuk memecahkan masalah bisnis relatif baru. Secara tradisional, mereka telah digunakan terutama dengan simulasi akademik. Namun, mengingat popularitas game dalam budaya kita, game bisnis sedang dikembangkan dan diterapkan untuk meningkatkan keterlibatan pelanggan dan karyawan. Gamification berkaitan dengan penerapan mekanika game ke situasi non-game. Gamification telah digunakan untuk mendesain ulang ruang kelas dan untuk mendukung pembelajaran, dan, seperti permainan, juga telah digunakan untuk meningkatkan keterlibatan pelanggan dan karyawan.

Baik dalam menggunakan game maupun gamification, rahasia kesuksesan berkaitan dengan memotivasi pelanggan dan/atau karyawan untuk tetap terlibat dengan proses bisnis. Meskipun pendekatan motivasi tradisional telah berhasil memotivasi karyawan di masa lalu, karena sifat dari perubahan jenis pekerjaan yang dilakukan, mereka tidak lagi berfungsi secara efisien atau efektif (lihat Bab 2). Pendekatan tradisional biasanya menggunakan pendekatan "tongkat dan wortel" untuk motivasi. Jika seorang individu (anak, siswa, atau karyawan) melakukan sesuatu yang tidak diinginkan, dia akan dihukum. Di sisi lain, jika mereka melakukan sesuatu yang diinginkan, mereka dihargai. Dengan kata lain, motivasi didasarkan sepenuhnya pada manfaat ekstrinsik. Game dan gamification berfokus terutama pada manfaat intrinsik; bukan manfaat ekstrinsik. Biasanya, Anda bermain game karena Anda menikmatinya. Kapan terakhir kali Anda bermain game karena Anda harus dan tidak ingin? Apakah Anda memainkannya untuk uang? Apakah itu agar Anda bisa menyenangkan orang lain? Atau, apakah itu agar Anda bisa menjadi bagian dari sesuatu yang lebih besar dari diri Anda sendiri? Seberapa menyenangkan itu? Apakah Anda termotivasi untuk memainkannya lagi? Mengapa? Kemungkinan uang bukanlah motivator yang cukup untuk memainkannya lagi. Tapi, bermain game untuk bersenang-senang mungkin akan memotivasi Anda untuk memainkannya berulang kali. Faktanya, tergantung pada tipe kepribadian game Anda, bermain game untuk menyenangkan orang lain atau menjadi bagian dari sesuatu yang lebih besar dari diri Anda sendiri dapat memotivasi Anda untuk memainkannya lagi.

Mengingat keberhasilan pengembangan game bisnis dan gamifikasi proses bisnis, ada beberapa hal yang dapat kita pelajari untuk meningkatkan antarmuka pengguna. Salah satu hal pertama adalah bahwa game dirancang secara eksplisit untuk menyenangkan. Biasanya, ketika kita merancang sistem informasi bisnis, salah satu hal terakhir yang kita pikirkan adalah apakah sistem itu menyenangkan untuk digunakan atau tidak. Kapan terakhir kali Anda menganggap menggunakan sistem informasi akuntansi sebagai hal yang menyenangkan? Namun, dalam hal ini, komponen menyenangkan dari sistem informasi bisnis berkaitan secara khusus dengan seberapa menarik antarmuka pengguna. Oleh karena itu, ada beberapa hal yang dapat kita terapkan dari game untuk mengembangkan antarmuka pengguna yang lebih menarik.

Pertama, game adalah tentang menciptakan pengalaman pengguna *(User Experience)*. Jelas, saat membuat pengalaman, perancang antarmuka pengguna *(User Interface)* harus

memperhatikan semua masalah yang telah kami jelaskan sebelumnya. Jika tidak, pengalaman tidak hanya akan menjadi ringan pada faktor keterlibatan, itu bisa menciptakan pengalaman negatif, bukan pengalaman positif yang kita harapkan.

Kedua, pengalaman permainan adalah semua tentang ide dan tema yang dijalin sepanjang permainan. Dalam kasus kami, ide bercerita (lihat Bab 3) dan Use-Case (lihat Bab 4) memberikan dasar untuk merancang dan mengembangkan pengalaman keterlibatan pengguna.

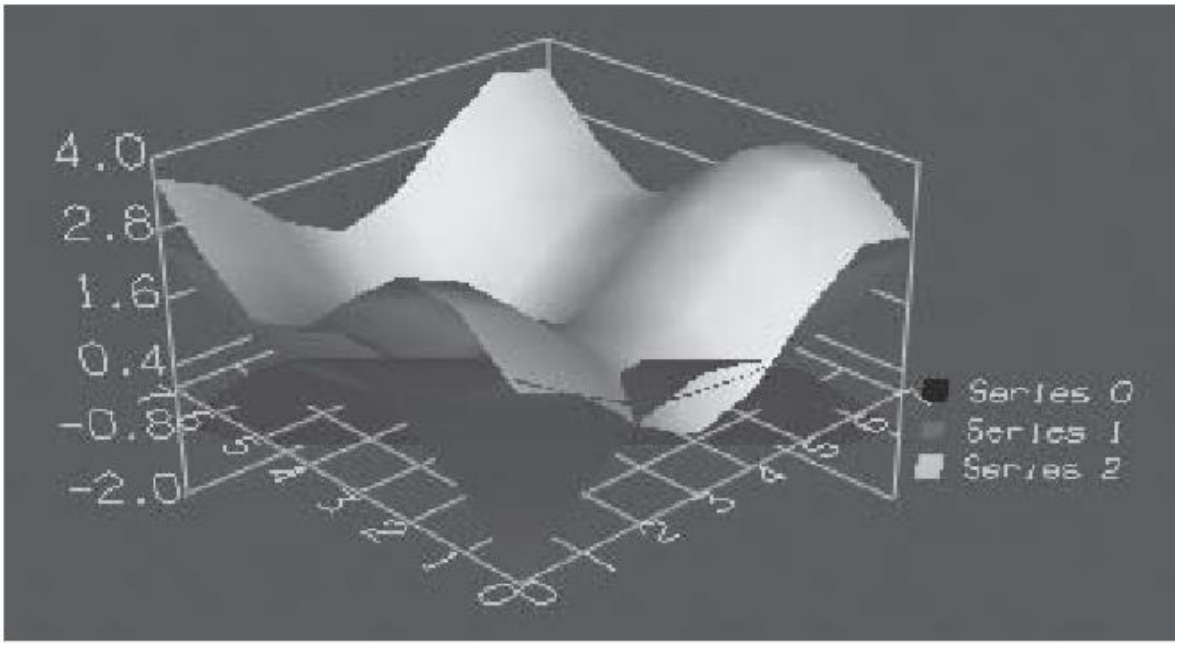
Ketiga, pengembang game sangat mengkhawatirkan pemain (pengguna). Ini membawa fokus yang lebih baik pada peran (aktor) yang dimainkan pengguna di sistem kami (lihat Bab 3). Ketika datang ke desain game, kita tidak hanya harus khawatir tentang tugas-tugas di mana pengguna akan terlibat, tetapi kita juga harus mulai berpikir tentang perbedaan psikologis dan kognitif individu di antara pengguna yang berbeda. Hal ini berlaku baik untuk berbagai jenis pelanggan dan karyawan.

Keempat, pengembang game juga cenderung mencoba dan membangun komunitas di sekitar game. Dengan cara ini, pengguna memiliki mekanisme dukungan bawaan. Schell30 menyarankan serangkaian tip untuk mengembangkan komunitas yang kuat yang dapat diterapkan pada pengembangan sistem informasi bisnis umum. Dia menyarankan bahwa kita harus membina persahabatan dengan mendorong pengguna untuk berbicara satu sama lain tentang sistem dan kita harus mencoba untuk menciptakan properti komunitas dengan meminta pengguna (dan pengembang) mengambil tanggung jawab bersama untuk sistem. Namun, salah satu sarannya yang lebih relevan adalah untuk mendukung berbagai tingkat pengguna berdasarkan tingkat pengalaman mereka. Dengan membuat sistem mendeteksi tingkat keahlian pengguna dalam menggunakan sistem, sistem dapat memperkenalkan fitur, seperti jalan pintas, setelah “tingkat” tertentu telah tercapai. Ini terkait dengan "naik level" dalam game. Ini akan membantu dalam mengatasi trade-off antara kemudahan belajar dan kemudahan penggunaan saat mengembangkan antarmuka pengguna. Ini juga dapat mendorong pengguna untuk "membeli" ke sistem.

Kelima, dalam hal desain game yang sukses, Anda harus mempertimbangkan estetika. Dengan demikian, tanpa fokus pada estetika, pengalaman yang Anda inginkan dari pelanggan atau karyawan mungkin kurang dari yang diinginkan. Bahkan, itu bisa membuat mereka enggan kembali ke situs Anda.

Desain Visualisasi Informasi Multidimensi

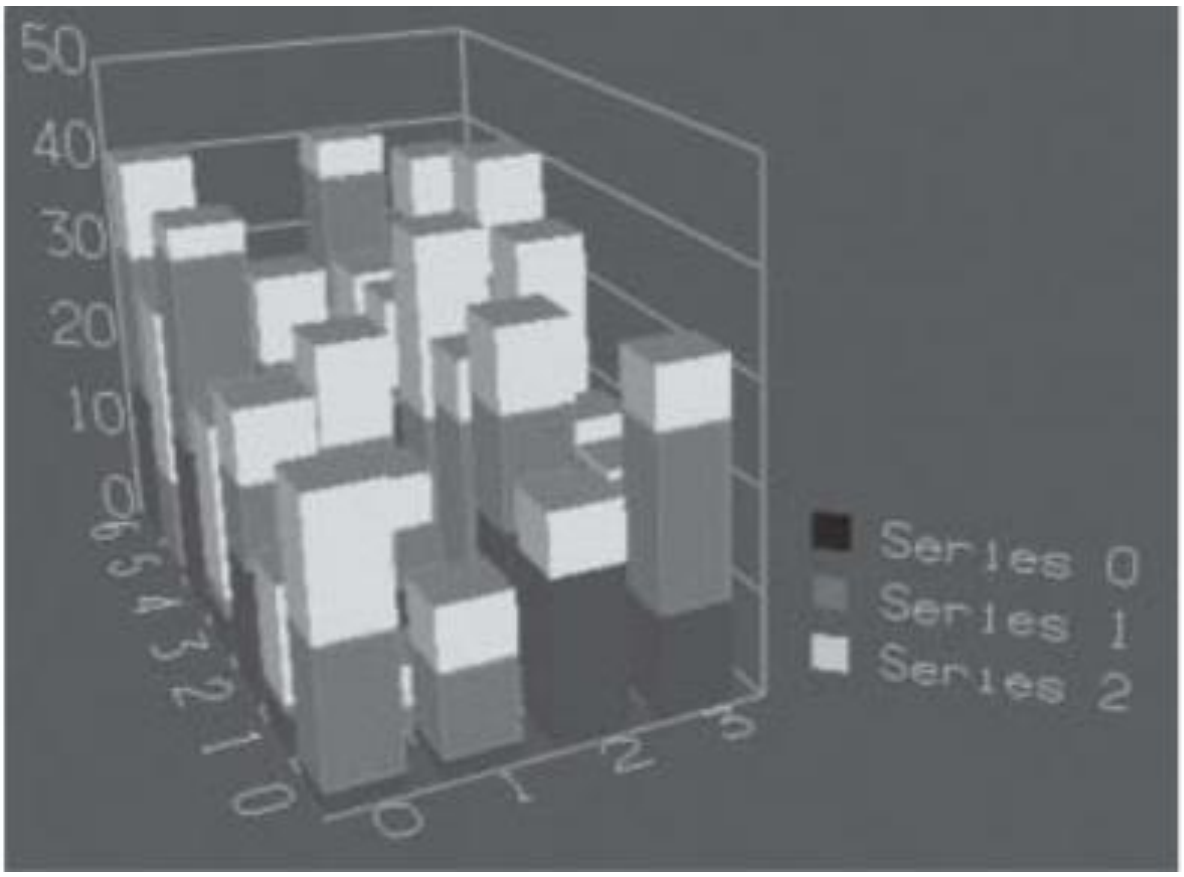
Ada banyak jenis visualisasi informasi multidimensi yang telah digunakan dalam bisnis. Namun, jenis yang berbeda jatuh ke dalam dua kategori dasar: visualisasi informasi multidimensi dalam ruang 2D dan visualisasi informasi multidimensi dalam ruang 3D nonimmersive. Visualisasi yang ditampilkan dalam ruang 2D mencakup bagan dan grafik bisnis dasar yang akan Anda temukan dalam spreadsheet atau paket statistik, misalnya, peta panas, peta, diagram tautan simpul, koordinat paralel, bagan radar, plot sebar, dan peta pohon. Masalah utama yang terkait dengan penggunaan jenis bagan dan diagram ini berkaitan dengan potensi bias yang merambat ke tampilan (lihat sebelumnya di bab ini). Namun, saat mempertimbangkan visualisasi yang ditampilkan dalam ruang 3D non-imersif, masalah tambahan akan muncul.



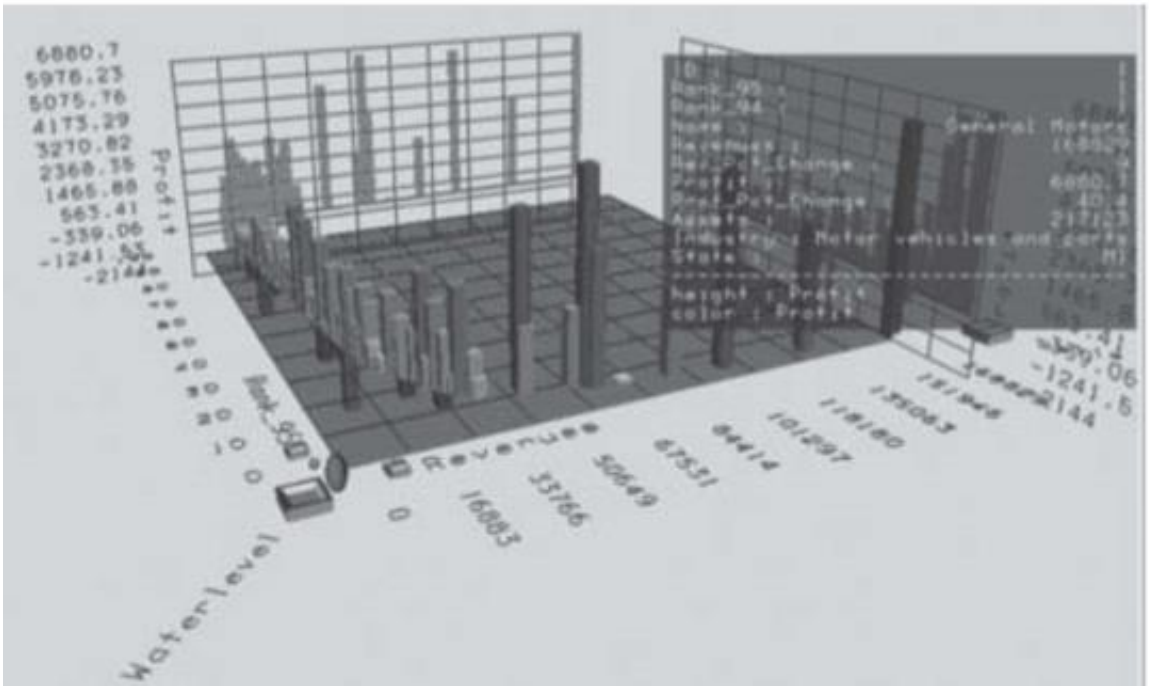
Gambar 10-17 Bagan Permukaan Multidimensi dalam Ruang 3D

Isu pertama yang muncul adalah persoalan mampu menentukan nilai tertentu yang direpresentasikan. Misalnya, pada Gambar 10-17, yang menggambarkan bagan permukaan multidimensi, hampir tidak mungkin untuk menentukan nilai spesifik yang direpresentasikan dalam ruang 3D. Dalam hal ini, ada empat nilai terpisah yang diplot: satu untuk sumbu X, satu untuk sumbu Y, satu untuk sumbu Z, dan satu untuk menggunakan warna permukaan. Hanya nilai terakhir, karena legenda, yang dapat dengan mudah ditentukan. Contoh lain dari masalah ini digambarkan pada Gambar 10-18 yang menunjukkan diagram batang multidimensi. Sekali lagi, empat nilai terpisah digambarkan. Ada dua pendekatan dasar yang digunakan untuk mengatasi masalah ini: kemampuan untuk memutar dan memperbesar visualisasi dan menyediakan kemampuan menelusuri yang memungkinkan nilai spesifik ditampilkan (lihat Gambar 10-19).

Masalah lain yang muncul saat menampilkan data dalam ruang 3D adalah oklusi; yaitu, saat melihat data dalam 3D, beberapa visualisasi mungkin ditutupi, disembunyikan, oleh bagian lain dari visualisasi. Misalnya, pada Gambar 10-18, nilai yang digambar di "belakang" visualisasi tidak dapat ditentukan dengan mudah. Pada Gambar 10-19, nilai negatif digambarkan di bawah permukaan lantai visualisasi. Dengan demikian, nilai-nilai negatif tidak dapat dilihat. Namun, dengan memutar visualisasi "naik" dan dapat mengklik nilai tertentu, nilai yang terkait dengan pengamatan itu dapat dibor ke dalam dan ditampilkan di jendela semi-transparan. Pada Gambar 10-20, visualisasi menampilkan nilai-nilai dasar di lantai. Dengan visualisasi ini, selain mendukung tampilan di dinding, pengguna juga dapat menggunakan bidang pengiris yang “memotong” visualisasi untuk membantu lebih memahami data yang divisualisasikan. Jenis visualisasi ini telah digunakan cukup luas dalam mendukung pengambilan keputusan bisnis.



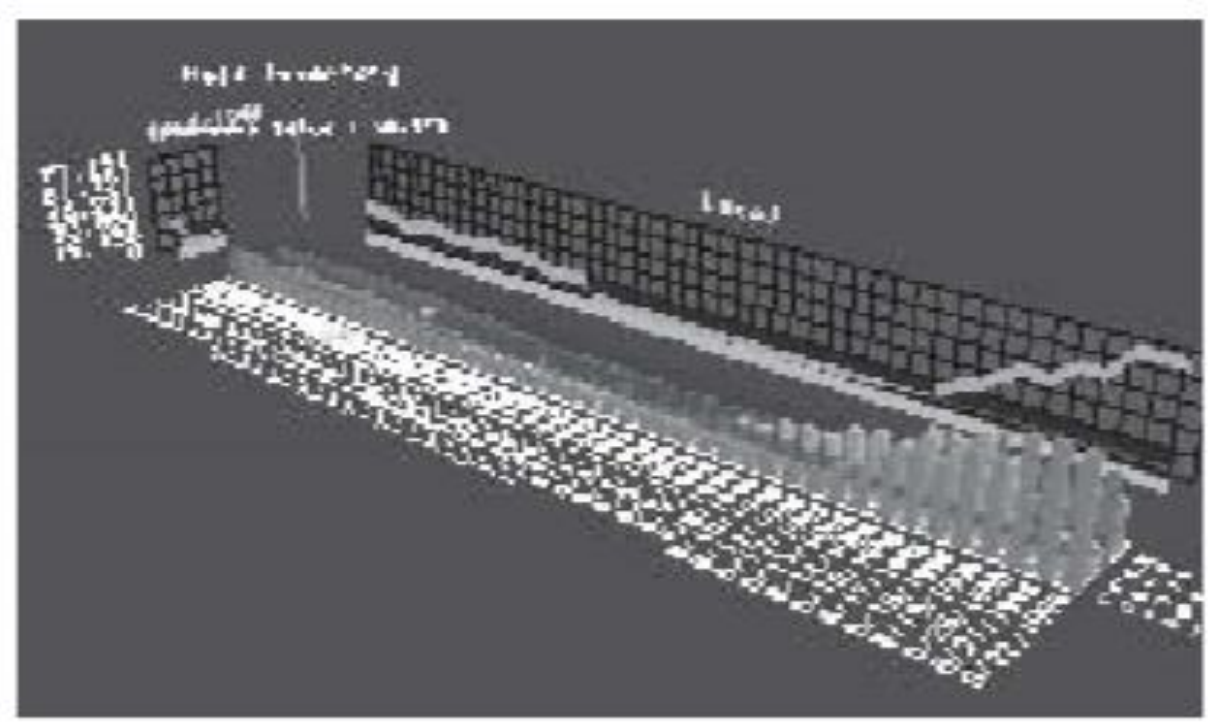
Gambar 10-18 Diagram Batang Multidimensi dalam Ruang 3D



Gambar 10-19 Diagram Batang Multidimensi pada Lantai dan Dinding dalam Ruang 3D

Ada lebih banyak jenis visualisasi informasi multidimensi yang telah digunakan dalam bisnis, misalnya, volume, lantai dan dinding, peta, dan permukaan. Masing-masing memiliki kekuatan, kelemahan, dan tantangannya sendiri. Selain itu, saat ini ada banyak alat khusus yang dapat digunakan untuk membantu merancang dan mengembangkan jenis visualisasi ini. Namun, proses desain dasar pada dasarnya adalah proses desain antarmuka pengguna yang

sama yang dijelaskan sebelumnya. Anda masih harus merancang kontrol navigasi, mekanisme input, dan output. Untuk memulainya, Anda harus memahami domain masalah yang mendasari dan tugas yang harus dilakukan pengguna. Selanjutnya, Anda harus memilih jenis visualisasi yang akan dirancang berdasarkan tugas yang perlu didukung oleh pengguna. Ini masih merupakan bentuk seni. Rekomendasi kami adalah duduk bersama pengguna dan melalui berbagai jenis visualisasi informasi untuk mencoba dan menentukan visualisasi informasi mana yang masuk akal. Keputusan ini harus didasarkan pada apakah pemetaan data ke visualisasi adalah "intuitif" dari perspektif pengguna atau tidak. Juga, ingat bahwa hanya karena Anda dapat menerapkan visualisasi informasi multidimensi yang kompleks tidak berarti Anda harus melakukannya. Dalam banyak kasus, bagan dan grafik bisnis sederhana sudah lebih dari cukup. Seperti desain game, pastikan untuk fokus pada estetika. Jika visualisasi tidak enak dipandang mata pengguna, maka mungkin visualisasi yang salah. Terakhir, mengingat bahwa desain visualisasi lebih merupakan seni daripada sains, pengujian keefektifan visualisasi dengan banyak pengguna sangatlah penting.



Gambar 10-20 Diagram Batang Multidimensi pada Lantai dengan Grafik Garis pada Dinding dalam Ruang 3D

Desain Antarmuka Pengguna dan Lingkungan Immersive

Augmented reality dan virtual menggunakan teknologi imersif, seperti Google Glass™ dan Oculus RiftTM, adalah salah satu area aplikasi terbaru dan menarik yang digunakan untuk memecahkan masalah bisnis. Dimana teknologi *virtual reality* (VR) benar-benar membenamkan pengguna ke dalam lingkungan digital simulasi buatan, teknologi *augmented reality* (AR) digunakan untuk menambah atau meningkatkan tampilan dunia nyata. Ada peluang dan tantangan dengan menerapkan kedua teknologi ini.

AR terutama digunakan dalam periklanan, navigasi dunia nyata, desain ruangan, game, jejaring sosial, dan aplikasi medis. Jika Anda kebetulan menonton NFL™ di TV, Anda telah mengalami augmented reality: Pikirkan garis "turun pertama" yang secara ajaib muncul di layar. Perangkat keras khas yang digunakan adalah smartphone atau tablet. Salah satu tantangan utama dalam AR adalah kemampuan menggunakan kamera untuk melihat dan

perangkat lunak untuk menafsirkan lanskap dunia nyata yang sedang ditambah oleh sistem. Ini dikenal sebagai pengenalan objek. Ini terutama menjadi masalah saat menggunakan browser AR yang "menghubungkan" dunia nyata dengan konten di Web. Dalam hal ini, browser, seperti JunaioTM, harus mengenali lokasi fisiknya dan menambahkan "tautan" untuk memungkinkan pengguna mencari informasi tambahan tentang lokasi yang dilihatnya. Untuk melakukan ini, browser harus menggunakan kamera, GPS, dan akselerometer smartphone. Isu utama lain yang berkaitan dengan AR adalah yang bersifat sosial. Misalnya, menggunakan Google Glass™ saat berbicara dengan seseorang dapat menimbulkan masalah tentang privasi, seperti apakah Anda memperhatikan orang lain atau apakah Anda melihat informasi yang ditampilkan dan apa sebenarnya informasi yang ditampilkan. Ada juga aplikasi yang tersedia saat ini yang mendukung pengenalan wajah, yang jelas menimbulkan lebih banyak masalah dengan privasi. Namun, ada berbagai kemungkinan manfaat dari penggunaan teknologi ini, misalnya, pikirkan tentang iklan yang terjadi di film Minority Report. Menggunakan Google Glass™ memungkinkan Anda melihat informasi yang telah dipersonalisasi tentang lokasi di sekitar Anda saat Anda berjalan di jalan.

Pendekatan prototyping, mirip dengan pendekatan desain antarmuka pengguna yang dijelaskan sebelumnya digunakan untuk merancang aplikasi AR. Semua masalah yang terkait dengan desain antarmuka pengguna tradisional, desain game, dan desain visualisasi informasi multidimensi relevan dan harus ditangani, misalnya, bayangkan menggunakan tampilan kepala saat mengemudikan mobil Anda untuk memberikan informasi tentang lokasi Anda; oklusi menjadi masalah nyata. Akibatnya, pengujian sistem AR dalam situasi dunia nyata sangat penting.

VR telah dilebih-lebihkan dan kurang dimanfaatkan. Meskipun hari ini VR terutama dikaitkan dengan game, itu telah digunakan dalam bisnis untuk waktu yang lama. Ini telah digunakan di berbagai bidang seperti perdagangan derivatif, manajemen risiko keuangan, kontrol proses industri, analisis pemasaran, pemodelan jaringan, manajemen operasi, pemodelan organisasi, manajemen portofolio, desain dan manufaktur produk, desain tata letak ruangan, analisis sensitivitas, rapat simulasi, stok analisis pasar, dan pelatihan. Namun, dari perspektif desain antarmuka pengguna, VR menimbulkan masalah tambahan yang perlu ditangani.

Diyakini bahwa VR mencapai kekuatannya dengan memikat perhatian pengguna dan mendorong rasa pendalaman, perasaan hadir di ruang yang disimulasikan. Tantangan menciptakan perasaan tenggelam memiliki dua dimensi utama: sensorik dan afektif. Dalam dimensi sensorik, kombinasi rangsangan yang digunakan harus cukup jelas sehingga proses persepsi otomatis individu dipicu, menghasilkan simulasi yang dianggap seperti kehidupan. Dalam dimensi afektif, pengguna harus berperan dalam peran eksplorasi yang interaktif. Ketika mempertimbangkan dimensi-dimensi ini, orang harus ingat bahwa realitas virtual berada dalam kesadaran individu dan, oleh karena itu, kontribusi relatif dari masing-masing dimensi ini dalam menciptakan rasa pencelupan akan bervariasi antar individu. Jadi, imersi adalah fungsi dari teknologi dan persepsi. Hal ini menimbulkan masalah perbedaan psikologis dan kognitif individu.

Interaksi dengan dunia virtual dapat berupa pencarian jalan melalui ruang virtual, menata ulang yang sudah ada, atau membuat objek 3D baru, atau berkomunikasi dengan agen lain (orang atau robot) yang berbagi ruang virtual yang sama. Pengunjung ke dunia maya yang besar seringkali tidak dapat memahami struktur topologi ruang secara keseluruhan. Mereka mungkin berkeliaran tanpa tujuan ketika mencoba menemukan lokasi tertentu untuk pertama kalinya dan selanjutnya mungkin mengalami kesulitan menemukan jalan kembali ke lokasi yang sudah dikunjungi. Tugas wayfinding mengharuskan pengguna untuk dapat mengkonseptualisasikan ruang virtual secara keseluruhan dan mengembangkan peta

kognitifnya. Sebuah peta kognitif tidak hanya terdiri dari hubungan spasial, tetapi juga kesan pendengaran, sensorik, dan emosional. Dalam permainan, peta biasanya disediakan untuk membantu memahami di mana seseorang berada di ruang virtual dan dari mana seseorang datang. Seperti masalah imersi, wayfinding juga mengangkat masalah yang berkaitan dengan perbedaan psikologis dan kognitif individu.

Tantangan terakhir terkait penggunaan VR sebagai platform antarmuka pengguna berkaitan dengan kolaborasi. Saat mempertimbangkan multiuser, sistem VR terdistribusi, seperti banyak video game saat ini, oklusi bisa menjadi masalah yang sangat besar. Objek di ruang VR tidak hanya dapat menyembunyikan objek VR lainnya, mereka juga dapat menyembunyikan pengguna lain. Juga, adalah mungkin bagi satu pengguna untuk melihat sesuatu yang menarik yang mungkin tidak dilihat oleh pengguna lain. Dalam hal ini, masalah wayfinding muncul kembali. Misalnya, jika pengguna A menemukan informasi yang menarik, maka pengguna A harus mengomunikasikan cara menavigasi ke lokasi di mana pengguna lain dapat mengamati temuan tersebut. Ada beberapa kemungkinan di sini, termasuk memberikan arah pencarian jalan dari "sudut pandang" tertentu, "menteleportasi" pengguna lain dari lokasi masing-masing saat ini ke lokasi pengguna A saat ini, meminta pengguna A mencari pengguna lain dan membawa mereka ke lokasi yang sesuai, atau meminta pengguna A untuk "mengarahkan" semua pengguna lain ke lokasi yang sesuai dengan mengambil alih kemampuan mereka untuk menavigasi melalui visualisasi. Selanjutnya, setelah pengguna lain berada di lokasi yang sesuai, bagaimana mereka kembali ke lokasi sebelumnya?

Jelas, merancang aplikasi VR yang efektif dan efisien sangat sulit. Sekali lagi, proses desain keseluruhan mirip dengan proses desain antarmuka pengguna umum yang dijelaskan sebelumnya. Namun, mengingat potensi VR untuk mendukung pengambilan keputusan bisnis dengan menggabungkan teknologi visualisasi game dan informasi ke dalam satu lingkungan terdistribusi yang mulus dan bahwa investasi dalam perangkat keras dan perangkat lunak khusus menurun, VR dapat memberikan hasil yang besar.

10.11 ISU INTERNASIONAL DAN BUDAYA DAN DESAIN ANTARMUKA PENGGUNA

Dengan World Wide Web, hampir semua perusahaan dapat memiliki kehadiran global. Dengan kemampuan ini, sebuah perusahaan harus menyadari serangkaian isu-isu internasional dan budaya. Isu-isu ini termasuk persyaratan multibahasa, warna, dan perbedaan budaya.

Persyaratan Multibahasa

Perbedaan pertama dan paling jelas antara aplikasi yang digunakan di satu wilayah dan yang dirancang untuk penggunaan global adalah bahasa. Aplikasi global sering kali memiliki persyaratan multibahasa, yang berarti bahwa mereka harus mendukung pengguna yang berbicara bahasa yang berbeda dan menulis menggunakan huruf non-Inggris (misalnya, mereka yang memiliki aksen, Sirilik, Jepang). Salah satu aspek yang paling menantang dalam merancang sistem global adalah mendapatkan terjemahan yang baik dari pesan bahasa asli ke dalam bahasa baru. Kata-kata sering kali memiliki arti yang serupa tetapi dapat menyampaikan arti yang agak berbeda ketika diterjemahkan, jadi penting untuk menggunakan penerjemah yang ahli dalam menerjemahkan kata-kata teknis. Beberapa aturan yang harus Anda ikuti adalah:

- Jaga agar tulisan tetap pendek dan sederhana. Jauh lebih mudah untuk menghindari kesalahan terjemahan.

- Humor, jargon, slang, klise, permainan kata, analogi, dan metafora yang hampa. Ini cenderung terlalu spesifik secara budaya. Akibatnya, poin mendasar yang dibuat kemungkinan besar akan hilang dalam terjemahan.
- Gunakan tata bahasa yang baik. Pastikan untuk menandai semuanya dengan benar. Meskipun Anda mungkin tergoda untuk mengabaikan aturan tata bahasa dan tanda baca untuk mencoba menjelaskan, itu membuat penerjemahan menjadi lebih sulit, terutama untuk sistem terjemahan otomatis. Jangan bergantung pada pemeriksa ejaan dan tata bahasa otomatis untuk menerapkan ini. Pada saat ini, mereka tidak cukup baik.

Tantangan lain sering kali adalah ruang layar. Secara umum, pesan berbahasa Inggris biasanya membutuhkan 20 persen hingga 30 persen lebih sedikit huruf daripada pesan berbahasa Prancis atau Spanyol. Merancang sistem global memerlukan pengalokasian lebih banyak ruang layar untuk pesan daripada yang mungkin digunakan dalam versi bahasa Inggris.

Beberapa sistem dirancang untuk menangani banyak bahasa dengan cepat sehingga pengguna di berbagai negara dapat menggunakan bahasa yang berbeda secara bersamaan; yaitu, sistem yang sama mendukung beberapa bahasa yang berbeda secara bersamaan (sistem multibahasa bersamaan). Sistem lain berisi bagian-bagian terpisah yang ditulis dalam setiap bahasa dan harus diinstal ulang sebelum bahasa tertentu dapat digunakan; yaitu, setiap bahasa disediakan oleh versi sistem yang berbeda sehingga setiap instalasi hanya akan menggunakan satu bahasa (yaitu, sistem multibahasa yang terpisah). Pendekatan mana pun bisa efektif, tetapi fungsionalitas ini harus dirancang ke dalam sistem dengan baik sebelum implementasi.

Terakhir, salah satu pertimbangan lain yang harus diperhatikan adalah arah bacaan. Di sebagian besar masyarakat Barat, pembaca membaca dari kiri ke kanan dan dari atas ke bawah. Ini tidak berlaku untuk banyak budaya. Misalnya, di negara-negara Arab, pembaca biasanya membaca dari kanan ke kiri dan dari atas ke bawah.

Warna

Untuk mulai dengan, warna tidak hitam dan putih. Makna yang terkait dengan warna sepenuhnya bergantung pada budaya. Faktanya, hitam dan putih belum tentu hitam dan putih; mereka bisa menjadi putih dan hitam. Di Indonesia sendiri, hitam dikaitkan dengan kematian, duka, dan kesedihan atau dengan rasa hormat dan formalitas. Misalnya, di Indonesia kami biasanya mengenakan pakaian hitam ke pemakaman, atau Anda akan berharap melihat para pemimpin agama berpakaian hitam (pikirkan tentang jubah yang biasanya dikenakan oleh seorang imam Katolik). Di banyak budaya Timur, di sisi lain, putih dikaitkan dengan kematian atau warna jubah yang dikenakan oleh para pemimpin agama. Dalam contoh yang dilaporkan oleh Singh dan Pereira, ketika warga lanjut usia di Amerika Serikat, Indonesia dan India diminta untuk "memvisualisasikan pernyataan berikut: Seorang wanita berpakaian putih, di tempat ibadah," hasil yang kembali hampir sama dengan berlawanan seperti yang bisa didapat. Di India, wanita itu akan menjadi janda, tetapi di Amerika Serikat dia diharapkan menjadi pengantin, sedangkan di Indonesia memiliki makna lain bisa jadi berbadah, atau suster gereja dan lain-lain.

Warna lain yang memiliki makna yang didorong oleh budaya termasuk hijau, biru, merah, kuning, dan ungu. Di Amerika Serikat, merah menyiratkan kegembiraan, gairah membumbui, seks, dan bahkan kemarahan; di Meksiko, itu menunjukkan agama; di Inggris, ini menunjukkan otoritas, kekuasaan, dan pemerintahan; di negara-negara Skandinavia, ini menunjukkan kekuatan; dan di Cina, itu berarti komunisme, kegembiraan, dan keberuntungan, sedangkan di Indonesia, warna merah menandakakan keberanian, semangat, nasionalisme, motivasi dan gairah. Biru dikaitkan dengan kekudusan di Israel; kebersihan di Skandinavia; cinta dan kebenaran di India; loyalitas di Jerman; dan kepercayaan, keadilan, dan bisnis

"resmi" di Amerika Serikat, dan menunjukkan dapat diandalkan dan bertanggung jawab sehingga dapat memberikan rasa aman dan dapat dipercaya. Di Irlandia, hijau menandakan nasionalisme dan Katolik, dan di Amerika Serikat itu menunjukkan kesehatan, lingkungan, keselamatan, keserakahan, dan kecemburuan. Di Arab Timur Tengah hijau melambangkan kesucian, di Prancis melambangkan kriminalitas, di Malaysia melambangkan bahaya dan penyakit, sedangkan di Indonesia hijau memiliki arti kesuburan, kedamaian, dan keagamaan. Kuning juga memiliki banyak makna yang bergantung pada budaya. Di Amerika Serikat, ini dikaitkan dengan kehati-hatian dan kepengecutan; di Skandinavia, kehangatan; di Jerman, iri; dan di India, perdagangan, di Indonesia warna kuning aidentik dengan kematian/duka. Ungu menandakan kematian, bangsawan, atau Gereja di Amerika Latin, Amerika Serikat, dan Italia, masing-masing. Jelas, ketika membangun situs web untuk audiens global, warna harus disetel dengan hati-hati; jika tidak, pesan yang tidak disengaja akan dikirim.

Perbedaan budaya

Kolumnis The New York Times Tom Friedman berbicara tentang perlunya sebuah perusahaan untuk menggunakan kemampuan lokalnya sendiri sebagai dasar untuk keunggulan kompetitif di pasar global. Ia menyebut proses ini sebagai glokalisasi. Dalam beberapa hal, ketika mengembangkan situs web untuk audiens internasional, Anda perlu mempertimbangkan kebalikan dari glokalisasi. Anda perlu memikirkan pesan apa yang perlu dikirim ke budaya lokal dari organisasi global Anda untuk mencapai tujuan bisnis perusahaan. Akibatnya, Anda harus mampu memahami budaya lokal yang berbeda. Isu-isu budaya telah dipelajari di tingkat organisasi dan nasional. Peneliti yang berbeda telah menekankan dimensi yang berbeda untuk memusatkan perhatian kita. Di bagian ini, kami membatasi diskusi kami pada masalah budaya yang memengaruhi perancangan antarmuka pengguna yang efektif. Secara khusus, kami hanya membahas penelitian Edward Hall dan Geert H ofstede.

Hall mengidentifikasi tiga dimensi yang secara langsung relevan dengan desain antarmuka pengguna: kecepatan pesan, konteks, dan waktu. Dimensi kecepatan pesan berkaitan dengan seberapa cepat seorang anggota budaya diharapkan untuk memahami pesan dan seberapa "dalam" isi pesan yang khas akan berada dalam budaya. Semakin dalam isi pesan, semakin lama waktu yang dibutuhkan anggota suatu budaya untuk memahami pesan tersebut. Misalnya, dua pendekatan berbeda untuk menggambarkan peristiwa sejarah adalah judul berita (cepat dan dangkal) dan film dokumenter (lambat dan dalam). Menurut Hall, budaya yang berbeda memiliki harapan yang berbeda dari isi dan respon terhadap pesan. Dimensi khusus ini berimplikasi pada isi pesan yang terkandung dalam antarmuka pengguna. Prinsip desain ketiga Krug ternyata didorong oleh budaya. Untuk audiens Barat, meminimalkan jumlah kata yang terkandung dalam antarmuka pengguna masuk akal. Orang Barat lebih suka langsung ke pokok permasalahan secepat mungkin. Namun, ini tidak berlaku untuk budaya Timur. Akibatnya, untuk perusahaan seperti Amazon.com, memberikan ulasan terperinci dan kutipan singkat dari sebuah buku memberikan dukungan untuk budaya yang lambat dan mendalam, sementara memberikan jenis komentar poin-poin mendukung budaya yang cepat dan dangkal. Dengan menyediakan keduanya, Amazon.com memenuhi kedua kebutuhan tersebut.

Dimensi kedua, konteks, berkaitan dengan tingkat informasi implisit yang digunakan dalam budaya versus informasi yang perlu dibuat eksplisit. Dalam budaya konteks tinggi, sebagian besar informasi diketahui secara intrinsik dan tidak harus dibuat eksplisit. Oleh karena itu, isi pesan yang sebenarnya cukup terbatas. Namun, dalam budaya konteks rendah, semuanya harus dijabarkan secara eksplisit untuk menghindari ambiguitas, dan oleh karena itu pesannya harus sangat rinci. Anda akan menemukan dimensi ini menyebabkan masalah ketika mencoba untuk menutup kesepakatan bisnis. Di sebagian besar masyarakat Barat, para pengacara ingin semuanya dijabarkan. Sebaliknya, di sebagian besar masyarakat Timur,

mungkin, pada kenyataannya, dianggap menghina jika harus mengeja semuanya. Dari perspektif desain situs web, Singh dan Pereira menunjukkan bahwa dalam budaya konteks tinggi, memfokuskan desain pada estetika, kesopanan, dan kerendahan hati menghasilkan situs web yang efektif, tetapi dalam budaya konteks rendah, hal-hal seperti syarat dan ketentuan pembelian, "peringkat" produk dan perusahaan, dan penggunaan superlatif dalam menggambarkan produk dan perusahaan adalah atribut penting dari situs web yang sukses.

Dimensi ketiga Hall, waktu, membahas bagaimana sebuah budaya berurusan dengan banyak hal berbeda yang terjadi secara bersamaan. Dalam budaya waktu polikronik, anggota budaya cenderung melakukan banyak hal pada waktu yang sama tetapi mudah teralihkan dan memandang komitmen waktu sebagai hal yang sangat fleksibel. Dengan budaya waktu monokronis, anggota budaya memecahkan banyak hal dengan berfokus pada satu hal pada satu waktu, berpikiran tunggal, dan mempertimbangkan komitmen waktu sebagai sesuatu yang ditetapkan. Saat mendesain untuk budaya polikronis, penggunaan pesan "pop-up" secara liberal mungkin menyenangkan dan menarik, sementara dalam budaya monokronis, pesan pop-up hanya mengganggu pengguna. Di masa lalu, budaya belahan bumi utara telah monokronik dan budaya belahan bumi selatan telah polikronik. Namun, dengan penggunaan interupsi email dan pesan teks, ini bisa berubah seiring waktu. Terlepas dari itu, membiarkan interupsi terjadi sebenarnya mengalihkan perhatian pengguna dari tugas mereka saat ini. Tergantung pada budayanya, ini bisa menjadi hal yang baik atau buruk untuk didukung.

Hofstede juga telah mengidentifikasi dimensi budaya yang relevan dengan antarmuka pengguna. Ini termasuk jarak kekuasaan, penghindaran ketidakpastian, individualisme versus kolektivisme, dan maskulinitas versus feminitas. Dimensi pertama, jarak kekuasaan, membahas bagaimana distribusi kekuasaan sosial ditangani dalam budaya. Dalam budaya dengan jarak kekuasaan yang tinggi, anggota budaya percaya pada otoritas hierarki sosial. Dalam budaya dengan jarak kekuasaan rendah, anggota budaya percaya bahwa kekuasaan harus lebih merata. Akibatnya, dalam budaya dengan jarak kekuasaan yang tinggi, penekanan pada "kebesaran" para pemimpin perusahaan, penggunaan "judul yang tepat" untuk anggota perusahaan, dan posting testimonial atas nama perusahaan dengan "menonjol" anggota masyarakat adalah penting. Penghargaan internasional yang dimenangkan oleh perusahaan, anggotanya, atau produknya juga harus dipasang secara mencolok di situs web.

Dimensi kedua, penghindaran ketidakpastian, membahas sejauh mana budaya nyaman dengan ketidakpastian. Dalam budaya dengan penghindaran ketidakpastian yang tinggi, anggota menghindari pengambilan risiko, menghargai tradisi, dan jauh lebih nyaman dalam masyarakat yang digerakkan oleh aturan. Dalam budaya yang mendapat skor tinggi pada penghindaran ketidakpastian, lebih banyak layanan pelanggan perlu disediakan, kontak "lokal" yang lebih penting harus tersedia, sejarah dan tradisi perusahaan dan produk perlu disediakan di situs web, dan, dalam kasus Software, penggunaan uji coba dan unduhan gratis sangat penting. Dengan kata lain, Anda perlu membangun kepercayaan dan mengurangi risiko yang dirasakan antara pelanggan dan perusahaan. Hal ini dapat didukung melalui segel persetujuan produk atau penggunaan sertifikasi WebTrust™ dan SysTrust™ untuk situs web.44 Hanya menerjemahkan situs web dari budaya penghindaran ketidakpastian rendah ke budaya penghindaran ketidakpastian tinggi tidaklah cukup. Anda juga perlu menunjukkan hubungan antara budaya lokal dan produk perusahaan.

Dimensi ketiga, individualisme versus kolektivisme, didasarkan pada tingkat penekanan budaya pada individu atau kolektif, atau kelompok. Di Amerika Utara dan Eropa, individualisme dihargai. Namun, di Asia Timur, diyakini bahwa dengan berfokus pada pengoptimalan kelompok, individu akan menjadi yang paling sukses. Dengan kata lain, kelompoklah yang paling penting. Dalam masyarakat kolektif, menyajikan informasi tentang bagaimana perusahaan “memberi kembali” kepada masyarakat; mendukung klub “anggota”,

program “loyalitas”, dan fasilitas “obrolan”; dan menyediakan tautan ke situs "lokal" yang menarik adalah karakteristik yang sangat penting untuk sebuah situs web. Sebaliknya, dalam masyarakat individualistis, memberikan dukungan untuk personalisasi pengalaman pengguna dengan situs web, menekankan keunikan produk yang dilihat pengguna, dan menekankan kebijakan privasi situs sangat penting.

Dimensi keempat Hofstede, maskulinitas versus feminitas, tidak berarti bagaimana laki-laki dan perempuan diperlakukan oleh budaya. Namun, dimensi ini justru membahas seberapa baik karakteristik maskulin dan feminin dinilai oleh budaya. Misalnya, dalam budaya maskulin, karakteristik seperti tegas, ambisius, agresif, dan kompetitif dihargai, sedangkan dalam budaya feminin, karakteristik seperti mendorong, penuh kasih, bijaksana, lembut, dan kooperatif dihargai. Dalam budaya maskulin, fokus pada efektivitas produk perusahaan sangat penting. Juga, memisahkan dengan jelas topik yang berorientasi pada pria dan wanita dan menempatkannya di bagian yang berbeda dari sebuah situs web dapat menjadi penting. Menurut Singh dan Pereira, budaya feminin menghargai fokus pada estetika dan menggunakan lebih banyak pendekatan soft-sell, di mana fokus pada aspek yang lebih afektif dan tidak berwujud dari perusahaan, anggotanya, dan produknya lebih tepat.

Jelas, mengoperasionalkan dimensi Hall dan Hofstede untuk desain antarmuka pengguna yang efektif tidaklah mudah. Selain itu, mengingat semua platform yang berbeda di mana antarmuka pengguna dapat digunakan, tingkat kerumitan dan kesulitan dalam merancang antarmuka pengguna yang efektif dan efisien yang mempertimbangkan dunia global dan multikultural tempat kita tinggal semakin meningkat. Namun, di pasar global, mengabaikan masalah budaya dalam desain antarmuka pengguna, apakah itu untuk sistem internal yang hanya digunakan oleh karyawan perusahaan atau sistem eksternal yang digunakan oleh pelanggan, pasti akan menyebabkan sistem gagal. Ini terutama benar ketika Anda mempertimbangkan situs media sosial dan seluler.

10.12 PERSYARATAN NON FUNGSIONAL DAN DESAIN LAPISAN INTERAKSI MANUSIA-KOMPUTER

Lapisan interaksi manusia-komputer sangat dipengaruhi oleh kebutuhan nonfungsional. Dalam bab ini, kami membahas masalah seperti tata letak antarmuka pengguna, kesadaran akan konten, estetika, pengalaman pengguna, dan konsistensi. Kami juga telah memberikan informasi tentang bagaimana merancang navigasi, input, dan output dari antarmuka pengguna. Terakhir, kami telah mempertimbangkan komputasi seluler, media sosial, lingkungan imersif dan multidimensi, serta masalah internasional dan budaya dalam desain antarmuka pengguna. Tak satu pun dari ini ada hubungannya dengan persyaratan fungsional sistem. Namun, jika diabaikan, sistem dapat menjadi tidak dapat digunakan. Seperti halnya lapisan manajemen data, ada empat jenis utama persyaratan nonfungsional yang penting dalam merancang lapisan interaksi manusia-komputer: persyaratan operasional, kinerja, keamanan, dan budaya dan politik.

Persyaratan operasional, seperti pilihan platform perangkat keras dan perangkat lunak, mempengaruhi desain lapisan interaksi manusia-komputer. Misalnya, sesuatu yang sederhana seperti jumlah tombol pada mouse (satu, dua, tiga, atau lebih) mengubah interaksi yang akan dialami pengguna. Persyaratan nonfungsional operasional lainnya yang dapat mempengaruhi desain lapisan interaksi manusia-komputer termasuk integrasi sistem dan portabilitas. Dalam kasus ini, solusi berbasis Web mungkin diperlukan, yang dapat mempengaruhi desain; tidak semua fitur antarmuka pengguna dapat diimplementasikan secara efisien dan efektif di Web. Ini dapat memerlukan desain antarmuka pengguna tambahan. Jelas, seluruh area komputasi seluler dapat mempengaruhi keberhasilan atau kegagalan sistem.

Persyaratan kinerja, dari waktu ke waktu, menjadi kurang menjadi masalah untuk lapisan ini. Namun, persyaratan kecepatan masih sangat penting, terutama dengan komputasi seluler. Sebagian besar pengguna tidak peduli untuk memukul kembali atau mengklik mouse dan harus istirahat sejenak sambil menunggu sistem untuk merespons, sehingga masalah efisiensi harus tetap ditangani. Tergantung pada toolkit antarmuka pengguna yang digunakan, komponen antarmuka pengguna yang berbeda mungkin diperlukan. Selanjutnya, interaksi lapisan interaksi manusia-komputer dengan lapisan lainnya harus diperhatikan. Misalnya, jika respons sistem lambat, menggabungkan struktur data yang lebih efisien dengan lapisan domain masalah, termasuk indeks dalam tabel dengan lapisan manajemen data, dan/atau mereplikasi objek di seluruh lapisan arsitektur fisik mungkin diperlukan.

Persyaratan keamanan yang mempengaruhi lapisan interaksi manusia-komputer terutama berkaitan dengan kontrol akses yang diterapkan untuk melindungi objek dari akses yang tidak sah. Sebagian besar kontrol ini ditegakkan melalui DBMS pada lapisan manajemen data dan sistem operasi pada lapisan arsitektur fisik. Namun, desain lapisan interaksi manusia-komputer harus mencakup kontrol log-on yang sesuai dan kemungkinan enkripsi.

Selain masalah internasional dan budaya yang dijelaskan sebelumnya, norma yang tidak disebutkan mempengaruhi persyaratan budaya dan politik yang dapat mempengaruhi desain lapisan interaksi manusia-komputer. Persyaratan norma yang tidak disebutkan termasuk menampilkan tanggal dalam format yang sesuai (MM/DD/YYYY versus DD/MM/YYYY). Agar sistem benar-benar berguna dalam lingkungan global, antarmuka pengguna harus dapat disesuaikan untuk memenuhi persyaratan budaya lokal.

Pertanyaan

1. Jelaskan tiga prinsip desain antarmuka pengguna yang penting.
2. Apa tiga bagian mendasar dari sebagian besar antarmuka pengguna?
3. Mengapa kesadaran konten itu penting?
4. Apa itu ruang putih, dan mengapa itu penting?
5. Dalam keadaan apa kepadatan harus rendah atauTinggi?
6. Bagaimana sebuah sistem dapat dirancang untuk digunakan oleh pengguna yang berpengalaman dan pengguna pertama kali?
7. Mengapa konsistensi dalam desain itu penting? Mengapa terlalu banyak konsistensi dapat menyebabkan masalah?
8. Bagaimana bagian yang berbeda dari antarmuka bisa konsisten?
9. Jelaskan proses dasar desain antarmuka pengguna.
10. Apa itu use case, dan mengapa itu penting?
11. Apa itu WND, dan mengapa digunakan?
12. Mengapa standar antarmuka penting?
13. Jelaskan tujuan dan isi metafora antarmuka, objek antarmuka, tindakan antarmuka, ikon antarmuka, dan templat antarmuka.
14. Mengapa kami membuat prototipe desain antarmuka pengguna?
15. Mengapa penting untuk melakukan evaluasi antarmuka sebelum sistem dibangun?
16. Bandingkan dan kontraskan empat jenis evaluasi antarmuka.
17. Dalam kondisi apa evaluasi heuristik dibenarkan?
18. Apa tiga prinsip desain Krug?
19. Jelaskan tiga prinsip dasar desain navigasi.
20. Bagaimana Anda bisa mencegah kesalahan?
21. Jelaskan perbedaan antara urutan objek-tindakan dan urutan tindakan-objek.
22. Jelaskan empat jenis kontrol navigasi
23. Mengapa menu merupakan kontrol navigasi yang paling umum digunakan?
24. Bandingkan dan kontraskan empat jenis menu.

25. Dalam situasi apa Anda akan menggunakan menu tarik-turun versus menu tab?
26. Dalam keadaan apa Anda akan menggunakan peta gambar versus menu daftar sederhana?
27. Jelaskan lima jenis pesan!
28. Apa faktor utama dalam merancang pesan kesalahan?
29. Apa itu bantuan peka konteks? Apakah pengolah kata Anda memiliki bantuan peka konteks?
30. Bagaimana Use-Case esensial dan Use-Case nyata berbeda?
31. Apa hubungan antara Use-Case penting dan skenario penggunaan?
32. Apa hubungan antara Use-Case nyata dan skenario penggunaan?
33. Jelaskan tiga prinsip dalam desain input.
34. Bandingkan dan kontraskan pemrosesan batch dan pemrosesan online. Jelaskan satu aplikasi yang akan menggunakan pemrosesan batch dan aplikasi yang akan menggunakan pemrosesan online.
35. Mengapa menangkap data pada sumbernya penting?
36. Jelaskan empat perangkat yang dapat digunakan untuk otomatisasi data sumber.
37. Jelaskan lima jenis input!
38. Mengapa validasi input penting?
39. Jelaskan lima jenis metode validasi input
40. Jelaskan tiga prinsip dalam desain output.
41. Jelaskan lima jenis keluaran.
42. Menurut Anda apa tiga kesalahan umum yang dilakukan analis pemula dalam desain navigasi?
43. Menurut Anda apa tiga kesalahan umum yang dilakukan analis pemula dalam desain input?
44. Menurut Anda apa tiga kesalahan umum yang dilakukan analis pemula dalam desain output?
45. Apa enam tantangan yang Anda hadapi saat mengembangkan aplikasi seluler?
46. Apa enam saran untuk mengatasi tantangan komputasi mobile?
47. Apa kontrol navigasi unik, mekanisme input, dan output yang didukung oleh komputasi seluler?
48. Berkenaan dengan media sosial, apa perbedaan antara pendekatan "dorong" dan "tarik" untuk berinteraksi dengan pelanggan?
49. Mengapa penting untuk menyinkronkan situs media sosial Anda?
50. Bagaimana Anda bisa membuat pelanggan Anda tetap terlibat dengan situs media sosial Anda?
51. Mengapa orang bermain game?
52. Apa itu gamifikasi?
53. Apa itu oklusi? Mengapa menjadi masalah ketika mengembangkan visualisasi informasi multidimensi? Sistem augmented reality? Sistem realitas virtual?
54. Apa itu augmented reality?
55. Sebutkan beberapa aplikasi bisnis potensial dari augmented reality.
56. Apa itu realitas maya?
57. Sebutkan beberapa aplikasi bisnis potensial dari virtual reality.
58. Saat mengembangkan sistem realitas virtual, apa saja masalah yang perlu ditangani?
59. Apa itu peta kognitif?
60. Apa saja masalah multibahasa yang mungkin Anda hadapi saat mengembangkan untuk audiens global?
61. Seberapa pentingkah penggunaan warna yang tepat saat mengembangkan situs web untuk audiens global? Berikan beberapa contoh potensi jebakan yang bisa Anda temui.
62. Sebutkan tiga dimensi budaya yang relevan dengan desain antarmuka pengguna yang diidentifikasi oleh Hall. Mengapa mereka relevan?

63. Sebutkan empat dimensi budaya yang relevan dengan desain antarmuka pengguna yang diidentifikasi oleh Hofstede. Mengapa mereka relevan?
64. Apa saja persyaratan nonfungsional yang dapat mempengaruhi desain lapisan interaksi manusia-komputer?

Latihan

A. Kembangkan dua skenario penggunaan untuk situs web yang menjual beberapa produk ritel (misalnya, buku, musik, pakaian).
B. Buat Storyboard untuk situs web yang menjual beberapa produk ritel (misalnya, buku, musik, pakaian).
C. Gambarlah WND untuk situs web yang menjual beberapa produk ritel (misalnya, buku, musik, pakaian).
D. Buat diagram tata letak jendela untuk halaman beranda situs web yang menjual beberapa produk ritel (mis., Buku, musik, pakaian).
E. Jelaskan komponen utama dari standar antarmuka untuk situs web yang menjual beberapa produk ritel (metafora, objek, tindakan, ikon, dan template).
F. Menggunakan Web, mengidentifikasi serangkaian permainan yang berguna dalam beberapa aspek bisnis, misalnya, periklanan atau pelatihan.
G. Menggunakan Web, mengidentifikasi satu set visualisasi informasi multidimensi yang digunakan untuk mendukung pengambilan keputusan bisnis.
H. Dengan menggunakan Web, temukan bisnis yang saat ini menggunakan sistem augmented reality dan virtual reality.
I. Untuk masalah A Real Estate Inc. di Bab 4 (latihan I, J, dan K), Bab 5 (latihan P dan Q), Bab 6 (latihan D), Bab 7 (latihan A), Bab 8 (latihan A) , dan Bab 9 (latihan L):
    1. Kembangkan dua skenario penggunaan.
    2. Menggambar WND.
    3. Desain Storyboard.
J. Berdasarkan solusi Anda untuk latihan I:
    1. Buat diagram tata letak windows untuk desain antarmuka.
    2. Kembangkan Use-Case nyata.
K. Untuk masalah A Video Store di Bab 4 (latihan L, M, dan N), Bab 5 (latihan R dan S), Bab 6 (latihan E), Bab 7 (latihan B), Bab 8 (latihan B), dan Bab 9 (latihan M):
    1. Kembangkan dua skenario penggunaan
    2. Menggambar WND.
    3. Desain Storyboard.
L. Berdasarkan solusi Anda untuk latihan K:
    1. Buat diagram tata letak windows untuk desain antarmuka.
    2. Kembangkan Use-Case nyata.
M. Untuk masalah keanggotaan gym di Bab 4 (latihan O, P, dan Q), Bab 5 (latihan T dan U), Bab 6 (latihan F), Bab 7 (latihan C), Bab 8 (latihan C), dan Bab 9 (latihan N):
    1. Kembangkan dua skenario penggunaan.
    2. Menggambar WND.
    3. Desain Storyboard.
N. Berdasarkan solusi Anda untuk latihan M:
    1. Buat diagram tata letak windows untuk desain antarmuka.
    2. Kembangkan Use-Case nyata.
O. Untuk soal Piknik R Us di Bab 4 (latihan R, S, dan T), Bab 5 (latihan V dan W), Bab 6 (latihan G), Bab 7 (latihan D), Bab 8 (latihan D), dan Bab 9 (latihan O):
    1. Kembangkan dua skenario penggunaan.
    2. Menggambar WND.
    3. Desain Storyboard.

P. Berdasarkan solusi Anda untuk latihan O:
    1. Buat diagram tata letak windows untuk desain antarmuka.
    2. Kembangkan Use-Case nyata.
Q. Untuk masalah Klub Bulanan di Bab 4 (latihan U, V, dan W), Bab 5 (latihan X dan Y), Bab 6 (latihan H), Bab 7 (latihan E), Bab 8 (latihan E), dan Bab 9 (latihan N):
    1. Kembangkan dua skenario penggunaan.
    2. Menggambar WND.
    3. Desain Storyboard.
R. Berdasarkan solusi Anda untuk latihan Q:
    1. Buat diagram tata letak windows untuk desain antarmuka.
    2. Kembangkan Use-Case nyata.
S. Buat desain antarmuka pengguna untuk solusi seluler untuk:
    1. Masalah Real Estat Inc.
    2. Masalah Toko Video.
    3. Masalah keanggotaan gym.
    4. Masalah Piknik R Us.
    5. Masalah Klub Bulanan.
T. Bagaimana jawaban Anda akan berubah untuk latihan I sampai S jika Anda berkembang untuk pasar global?

10.13 MINICASE

1. Baby Fashion adalah pengecer katalog yang mengkhususkan diri dalam pakaian anak-anak. Sebuah proyek telah berjalan untuk mengembangkan sistem entri pesanan baru untuk pegawai katalog perusahaan. Sistem lama memiliki antarmuka pengguna berbasis karakter yang sesuai dengan dasar COBOL sistem. Sistem baru ini akan menampilkan antarmuka pengguna grafis yang lebih sesuai dengan produk PC terkini yang digunakan saat ini. Perusahaan berharap bahwa antarmuka pengguna baru ini akan membantu mengurangi omset yang dialami oleh petugas entri pesanannya. Banyak staf entri pesanan yang baru direkrut merasa sistem lama sangat sulit untuk dipelajari dan kewalahan oleh banyak kode misterius yang harus digunakan untuk berkomunikasi dengan sistem.

   Evaluasi penelusuran antarmuka pengguna dijadwalkan hari ini untuk memberi pengguna pandangan pertama pada antarmuka sistem baru. Tim proyek berhati-hati untuk mengundang beberapa pengguna utama dari departemen entri pesanan. Secara khusus, Norma dimasukkan karena pengalamannya selama bertahun-tahun dengan sistem entri pesanan. Norma dikenal sebagai pemimpin informal di departemen tersebut; pendapatnya mempengaruhi banyak rekan-rekannya. Norma telah memberitahukan bahwa dia kurang senang dengan ide-ide yang dia dengar untuk sistem baru. Karena pengalaman dan ingatannya yang baik, Norma bekerja sangat efektif dengan sistem berbasis karakter dan mampu menyelesaikan transaksi yang paling rumit sekalipun dengan mudah. Norma kesulitan menahan cibiran ketika dia mendengar pembicaraan tentang hal-hal seperti "ikon" dan "tombol" di antarmuka pengguna baru.

   Cindy juga diundang ke walkthrough karena pengaruhnya di departemen entri pesanan. Cindy telah bekerja di departemen tersebut hanya selama satu tahun, tetapi dia dengan cepat menjadi terkenal karena keberhasilannya mengorganisasi layanan penitipan anak yang sakit untuk anak-anak pekerja departemen. Anak-anak yang sakit adalah penyebab nomor satu ketidakhadiran di departemen, dan banyak pekerja tidak mampu untuk melewatkan hari kerja. Tidak pernah tinggal diam ketika situasi membutuhkan perbaikan, Cindy telah menjadi pendukung vokal dari sistem baru.

a. Berdasarkan prinsip-prinsip desain yang disajikan dalam teks, jelaskan fitur antarmuka pengguna yang paling penting bagi pengguna berpengalaman seperti Norma.
b. Berdasarkan prinsip-prinsip desain yang disajikan dalam teks, jelaskan fitur antarmuka pengguna yang paling penting bagi pengguna pemula seperti Cindy.

2. Anggota tim proyek pengembangan sistem pergi makan siang bersama, dan seperti yang sering terjadi, percakapan berubah menjadi berhasil. Tim telah bekerja pada pengembangan desain antarmuka pengguna, dan sejauh ini, pekerjaan telah berjalan dengan lancar. Tim harus menyelesaikan pekerjaan pada prototipe antarmuka awal minggu depan. Kombinasi storyboard dan prototipe bahasa telah digunakan dalam proyek ini. Storyboard menggambarkan keseluruhan struktur dan aliran sistem, tetapi tim mengembangkan prototipe bahasa dari layar yang sebenarnya karena mereka merasa bahwa melihat layar yang sebenarnya akan berharga bagi pengguna.

   Ade (anggota termuda dari tim proyek): Saya membaca artikel tadi malam tentang cara yang sangat keren untuk mengevaluasi desain antarmuka pengguna. Ini disebut pengujian kegunaan, dan itu dilakukan oleh semua vendor perangkat lunak utama. Saya pikir kita harus menggunakannya untuk mengevaluasi desain antarmuka kita.

   Nia (analis sistem): Saya juga pernah mendengarnya, tetapi bukankah itu sangat mahal?

   Yakub (manajer proyek): Saya khawatir itu mahal dan saya tidak yakin kami dapat membenarkan biaya untuk proyek ini.

   Ade: Tapi kita benar-benar perlu tahu bahwa antarmuka berfungsi. Saya pikir teknik pengujian kegunaan ini akan membantu kami membuktikan bahwa kami memiliki desain yang bagus.

   Eri (analis sistem): Akan, Ade, tetapi ada cara lain juga. Saya berasumsi bahwa kami akan melakukan penelusuran menyeluruh dengan pengguna kami dan menyajikan antarmuka kepada mereka di rapat. Kami dapat memproyeksikan setiap layar antarmuka sehingga pengguna dapat melihatnya dan memberi kami reaksi mereka. Ini mungkin cara paling efisien untuk mendapatkan respons pengguna terhadap pekerjaan kami.

   Nia: Itu benar, tapi saya pasti ingin melihat pengguna duduk dan bekerja dengan sistem. Saya selalu belajar banyak dengan melihat apa yang mereka lakukan, melihat di mana mereka bingung, dan mendengar komentar dan umpan balik mereka.

   Ryan (analis sistem): Bagi saya, kami telah melakukan begitu banyak pekerjaan dalam desain antarmuka ini sehingga yang perlu kami lakukan hanyalah meninjaunya sendiri. Mari kita buat daftar prinsip-prinsip desain yang paling kita perhatikan dan periksa sendiri untuk memastikan kita telah mengikutinya secara konsisten. Jika kita punya, kita harus baik-baik saja. Kami ingin melanjutkan implementasinya, Anda tahu.

   Yakub: Ini semua adalah ide bagus. Sepertinya kita semua memiliki pandangan berbeda tentang cara mengevaluasi desain antarmuka. Mari kita coba memilah teknik yang terbaik untuk proyek kita.

   Kembangkan seperangkat pedoman yang dapat membantu tim proyek seperti ini memilih teknik evaluasi antarmuka yang paling tepat untuk proyek mereka.

3. Struktur menu untuk sistem berbasis karakter Holiday Travel Vehicle yang ada ditampilkan di sini. Mengembangkan dan membuat prototipe desain antarmuka baru

untuk fungsi sistem menggunakan antarmuka pengguna grafis. Juga, kembangkan satu set Use-Case nyata untuk antarmuka baru Anda. Asumsikan sistem baru perlu menyertakan fungsi yang sama seperti yang ditunjukkan pada menu yang disediakan. Sertakan pesan apa pun yang akan dihasilkan saat pengguna berinteraksi dengan antarmuka Anda (kesalahan, konfirmasi, status, dll.). Juga, siapkan ringkasan tertulis yang menjelaskan bagaimana antarmuka Anda mengimplementasikan prinsip-prinsip desain antarmuka yang baik seperti yang disajikan dalam buku teks.

Holiday Travel Vehicles

Main Menu

1. Faktur penjualan
2. Inventaris Kendaraan Vehicle
3. Laporan
4. Karyawan bagian penjualan

Ketik jumlah pilihan menu di sini:.....__

Holiday Travel Vehicles

Menu Faktur Penjualan

4. Buat Faktur Penjualan
5. Ubah Faktur Penjualan
6. Batalkan Faktur Penjualan

Ketik jumlah pilihan menu di sini:....._
_

Holiday Travel Vehicles

Menu Inventaris Kendaraan

4. Buat Catatan Inventaris Kendaraan
5. Ubah Catatan Inventaris Kendaraan
6. Hapus Catatan Inventaris Kendaraan

Ketik jumlah pilihan menu di sini:.....__

Holiday Travel Vehicles

Menu Laporan

3. Laporan Komisi

4. Penjualan RV dengan Membuat Laporan
5. Penjualan Trailer dengan Membuat Laporan
6. Laporan Opsi Dealer

Ketik jumlah pilihan menu di sini:.....__

Holiday Travel Vehicles

Menu Pemeliharaan Staf Penjualan

65. Tambahkan Catatan Tenaga Penjual
66. Ubah Catatan Tenaga Penjual
67. Hapus Catatan Tenaga Penjual

Ketik jumlah pilihan menu di sini:.....__

4. Salah satu aspek dari sistem baru yang sedang dikembangkan di Holiday Travel Vehicles adalah entri langsung faktur penjualan ke dalam sistem komputer oleh tenaga penjual saat transaksi pembelian sedang diselesaikan. Dalam sistem saat ini, tenaga penjual mengisi formulir kertas (ditampilkan di halaman berikutnya).

   Merancang dan membuat prototipe layar input yang memungkinkan wiraniaga memasukkan semua informasi yang diperlukan untuk faktur penjualan. Informasi berikut mungkin berguna dalam proses desain Anda. Asumsikan bahwa Holiday Travel Vehicles menjual kendaraan rekreasi dan trailer dari empat pabrikan yang berbeda. Setiap pabrikan memiliki sejumlah nama dan model RV dan trailer yang tetap.

   Untuk keperluan prototipe Anda, gunakan format ini:

   Mfg-A Name-1 Model-X

   Mfg-A Name-1 Model-Y

   Mfg-A Name-1 Model-Z

   Mfg-B Name-1 Model-X

   Mfg-B Name-1 Model-Y

   Mfg-B Name-2 Model-X

   Mfg-B Name-2 Model-Y

   Mfg-B Name-2 Model-Z

   Mfg-C Name-1 Model-X

Mfg-C Name-1 Model-Y

Mfg-C Name-1 Model-Z

Mfg-C Name-2 Model-X

Mfg-C Name-3 Model-X

Mfg-D Name-1 Model-X

Mfg-D Name-2 Model-X

Mfg-D Name-2 Model-Y

Juga, asumsikan ada sepuluh opsi dealer berbeda yang dapat dipasang pada kendaraan atas permintaan pelanggan. Perusahaan saat ini memiliki sepuluh staf penjualan.

Holiday Travel Vehicles
Sales Invoice

Invoice #: _____
Invoice Date: _____

Customer Name: _____
Address: _____
City: _____
State: _____
Zip: _____
Phone: _____

New RV/TRAILER
(circle one)
Name: _____
Model: _____
Serial #: _____ Year: _____
Manufacturer: _____

Trade-in RV/TRAILER
(circle one)
Name: _____
Model: _____
Year: _____
Manufacturer: _____

Options:

Code	Description	Price
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

Vehicle Base Cost: _____
Trade-in Allowance: _____
Total Options: _____
Tax: _____
License Fee: _____
Final Cost: _____

_____
(Salesperson Name)

_____
(Customer Signature)

5. Lihat Minicase Professional and Scientific Staff Management (PSSM) di Bab 4, 6, 7, 8, dan 9.
   a. Kembangkan dua skenario penggunaan, gambar WND, dan rancang storyboard.

b. Berdasarkan jawaban Anda untuk bagian a, buat diagram tata letak jendela untuk antarmuka pengguna dan kembangkan satu set Use-Case nyata untuk antarmuka pengguna.
c. Bagaimana desain antarmuka pengguna Anda harus dimodifikasi jika Anda ingin menerapkannya di tablet? Bagaimana dengan smartphone?
d. Apa, jika ada, situs media sosial yang harus dipertimbangkan PSSM?
e. Bagaimana jawaban Anda akan berubah jika Anda mengembangkan sistem untuk audiens global?

BAB 11

DESAIN LAPISAN ARSITEKTUR FISIK

Komponen penting dari desain sistem informasi adalah desain lapisan arsitektur fisik, yang menggambarkan perangkat keras, perangkat lunak, dan lingkungan jaringan sistem. Desain lapisan arsitektur fisik mengalir terutama dari persyaratan nonfungsional, seperti persyaratan operasional, kinerja, keamanan, budaya, dan politik. Hasil dari desain lapisan arsitektur fisik meliputi arsitektur dan spesifikasi perangkat keras dan perangkat lunak.

11.1 Tujuan

- Memahami berbagai komponen arsitektur fisik.
- Memahami arsitektur fisik berbasis server, berbasis klien, dan klien-server.
- Pahami komputasi awan, komputasi di mana-mana, dan Internet of things (IoT), dan Green IT.
- Mampu membuat model jaringan menggunakan diagram penyebaran.
- Pahami cara membuat spesifikasi perangkat keras dan perangkat lunak.
- Memahami bagaimana persyaratan operasional, kinerja, keamanan, budaya, dan politik memengaruhi desain lapisan arsitektur fisik.

11.2 Pendahuluan

Di lingkungan saat ini, sebagian besar sistem informasi tersebar di banyak komputer. Sebuah sistem berbasis Web, misalnya, berjalan di browser pada komputer desktop tetapi berinteraksi dengan server Web (dan mungkin komputer lain) melalui Internet. Sebuah sistem yang beroperasi sepenuhnya di dalam jaringan perusahaan mungkin memiliki program Visual Basic yang diinstal pada satu komputer tetapi berinteraksi dengan server database di tempat lain di jaringan. Oleh karena itu, langkah desain yang penting adalah pembuatan desain lapisan arsitektur fisik, rencana bagaimana sistem akan didistribusikan ke seluruh komputer, dan perangkat keras dan perangkat lunak apa yang akan digunakan untuk setiap komputer.

Dalam banyak kasus, sistem dibangun untuk menggunakan perangkat keras dan perangkat lunak yang ada dalam organisasi. Oleh karena itu, arsitektur saat ini membatasi pilihan. Faktor lain seperti standar perusahaan, perjanjian lisensi situs yang ada, dan hubungan produk-vendor juga dapat mengamanatkan arsitektur, perangkat keras, dan perangkat lunak apa yang harus digunakan oleh tim proyek. Namun, banyak organisasi sekarang memiliki berbagai infrastruktur yang tersedia atau secara terbuka mencari proyek percontohan untuk menguji arsitektur baru yang memungkinkan tim proyek untuk memilih satu berdasarkan faktor penting lainnya.

Merancang lapisan arsitektur fisik bisa sangat sulit; oleh karena itu, banyak organisasi mempekerjakan konsultan ahli atau menugaskan analis yang sangat berpengalaman untuk tugas tersebut. Dalam bab ini, kita memeriksa faktor utama dalam desain lapisan arsitektur fisik, tetapi penting untuk diingat bahwa dibutuhkan banyak pengalaman untuk melakukannya dengan baik. Persyaratan nonfungsional yang dikembangkan selama analisis (lihat Bab 3) memainkan peran utama dalam desain lapisan arsitektur fisik. Persyaratan ini diperiksa ulang dan disempurnakan menjadi persyaratan yang lebih rinci yang mempengaruhi arsitektur sistem.

11.3 ELEMEN LAPISAN ARSITEKTUR FISIK

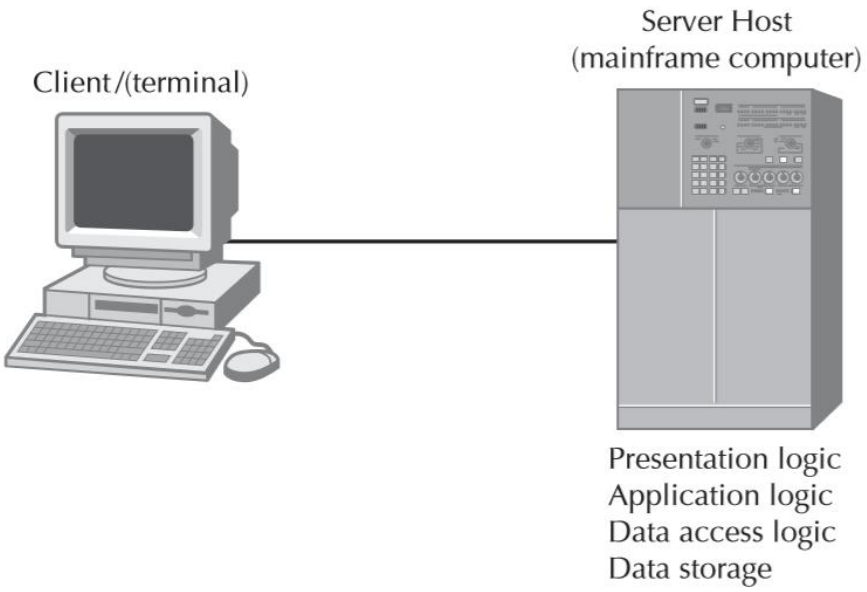
Tujuan merancang lapisan arsitektur fisik adalah untuk menentukan bagian mana dari perangkat lunak aplikasi yang akan ditugaskan ke perangkat keras apa. Meskipun ada banyak cara komponen perangkat lunak dapat ditempatkan pada komponen perangkat keras, ada tiga arsitektur aplikasi utama yang digunakan saat ini: arsitektur berbasis server, arsitektur berbasis klien, dan arsitektur klien-server.

Komponen Arsitektur

Komponen arsitektur utama dari sistem apapun adalah perangkat lunak dan perangkat keras. Komponen perangkat lunak utama dari sistem yang sedang dikembangkan harus diidentifikasi dan kemudian dialokasikan ke berbagai komponen perangkat keras di mana sistem akan beroperasi. Masing-masing komponen ini dapat digabungkan dalam berbagai cara yang berbeda.

Semua sistem perangkat lunak dapat dibagi menjadi empat fungsi dasar. Yang pertama adalah penyimpanan data (terkait dengan persistensi objek yang terletak di lapisan manajemen data—lihat Bab 9). Sebagian besar program aplikasi memerlukan data untuk disimpan dan diambil kembali, baik informasi tersebut berupa file kecil seperti memo yang dihasilkan oleh pengolah kata atau database besar yang menyimpan catatan akuntansi organisasi. Ini adalah data yang didokumentasikan dalam model struktural (kartu CRC dan diagram kelas). Fungsi kedua adalah logika akses data (terkait dengan kelas akses dan manipulasi data yang terletak di lapisan manajemen data—lihat Bab 9), pemrosesan yang diperlukan untuk mengakses data, yang sering kali berarti kueri database dalam SQL (bahasa kueri terstruktur). Fungsi ketiga adalah logika aplikasi (terletak pada lapisan domain masalah—lihat Bab 4 sampai 8), yang bisa sederhana atau kompleks, tergantung pada aplikasinya. Ini adalah logika yang didokumentasikan dalam fungsional (diagram aktivitas dan Use-Case) dan model perilaku (urutan, komunikasi, dan mesin status perilaku). Fungsi keempat adalah logika presentasi (terletak pada lapisan interaksi manusia-komputer—lihat Bab 10), penyajian informasi kepada pengguna, dan penerimaan perintah pengguna (antarmuka pengguna). Keempat fungsi ini (penyimpanan data, logika akses data, logika aplikasi, dan logika presentasi) adalah blok bangunan dasar dari aplikasi apa pun.

Tiga komponen perangkat keras utama dari suatu sistem adalah komputer klien, server, dan jaringan yang menghubungkannya. Komputer klien adalah perangkat input/output yang digunakan oleh pengguna dan biasanya komputer desktop atau laptop, tetapi juga dapat berupa perangkat genggam, telepon seluler, terminal tujuan khusus, dan sebagainya. Server biasanya komputer yang lebih besar yang digunakan untuk menyimpan perangkat lunak dan perangkat keras yang dapat diakses oleh siapa saja yang memiliki izin. Jaringan yang menghubungkan komputer dapat bervariasi dalam kecepatan dari ponsel yang lambat, ke jaringan frame relai yang selalu aktif dengan kecepatan sedang, hingga koneksi broadband yang selalu aktif dengan cepat seperti modem kabel, DSL, atau sirkuit T1, hingga selalu berkecepatan tinggi. - di sirkuit ethernet, T3, atau ATM.



Gambar 11-1 Arsitektur Berbasis Server

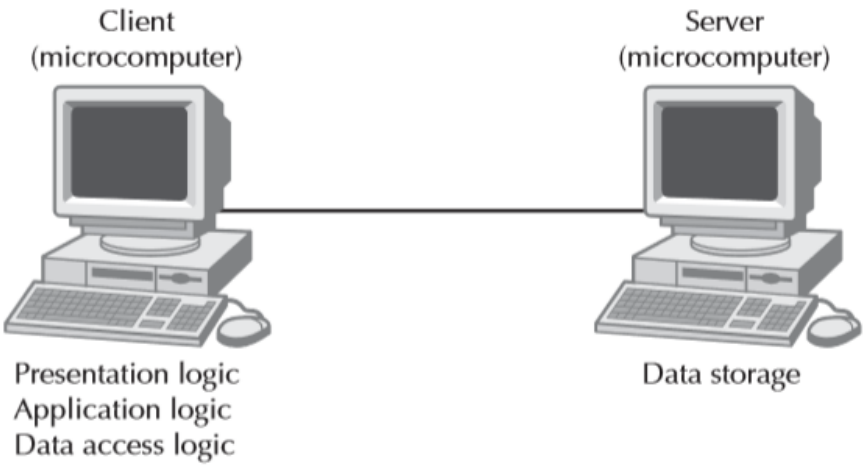
Arsitektur Berbasis Server

Arsitektur komputasi pertama adalah arsitektur berbasis server, dengan server melakukan keempat fungsi. Klien memungkinkan pengguna untuk mengirim dan menerima pesan ke dan dari server. Klien hanya menangkap penekanan tombol dan mengirimkannya ke server untuk diproses dan menerima instruksi dari server tentang apa yang akan ditampilkan (lihat Gambar 11-1).

Arsitektur yang sangat sederhana ini seringkali bekerja dengan sangat baik. Perangkat lunak aplikasi dikembangkan dan disimpan di satu komputer, dan semua data ada di komputer yang sama. Ada satu titik kendali, karena semua pesan mengalir melalui satu server pusat. Masalah mendasar dengan jaringan berbasis server adalah bahwa server harus memproses semua pesan. Seiring dengan meningkatnya permintaan aplikasi, banyak komputer server menjadi kelebihan beban dan tidak dapat dengan cepat memproses semua permintaan pengguna. Waktu respons menjadi lebih lambat, dan pengelola jaringan harus mengeluarkan lebih banyak uang untuk meningkatkan komputer server. Sayangnya, upgrade datang dalam jumlah besar dan mahal; sulit untuk meningkatkan "sedikit".

Arsitektur Berbasis Klien

Dengan arsitektur berbasis klien, klien adalah komputer pribadi di jaringan area lokal (LAN), dan komputer server adalah server di jaringan yang sama. Perangkat lunak aplikasi pada komputer klien bertanggung jawab atas logika presentasi, logika aplikasi, dan logika akses data; server hanya menyimpan data (lihat Gambar 11-2).



Gambar 11-2 Arsitektur Berbasis Klien

Arsitektur sederhana ini juga sering bekerja dengan baik. Namun, karena tuntutan akan semakin banyak aplikasi jaringan tumbuh, sirkuit jaringan dapat menjadi kelebihan beban. Masalah mendasar dalam jaringan berbasis klien adalah bahwa semua data di server harus dikirim ke klien untuk diproses. Misalnya, pengguna ingin menampilkan daftar semua karyawan dengan asuransi jiwa perusahaan. Semua data dalam database harus berjalan dari server tempat database disimpan melalui jaringan ke klien, yang kemudian memeriksa setiap record untuk melihat apakah cocok dengan data yang diminta oleh pengguna. Ini dapat membebani jaringan dan kekuatan komputer klien.

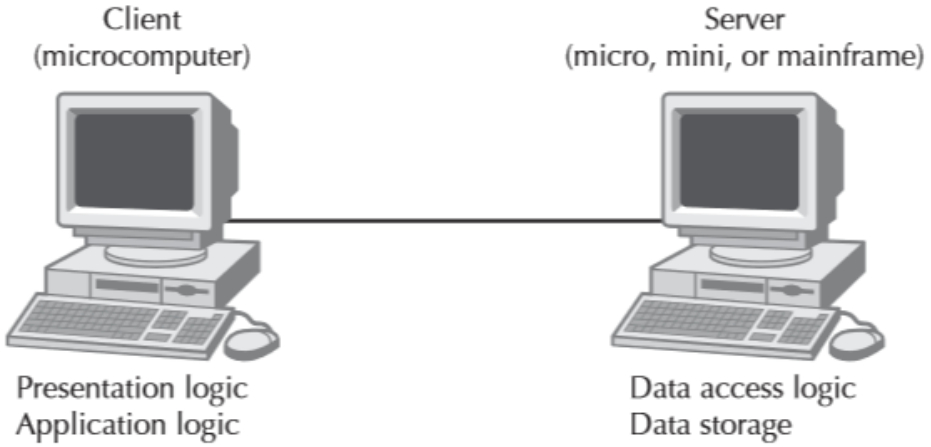
Arsitektur Klien–Server

Sebagian besar organisasi saat ini menggunakan arsitektur client-server, yang berusaha menyeimbangkan pemrosesan antara klien dan server dengan meminta keduanya melakukan beberapa fungsi aplikasi. Dalam arsitektur ini, klien bertanggung jawab atas logika presentasi, sedangkan server bertanggung jawab atas logika akses data dan penyimpanan data. Logika aplikasi dapat berada di klien atau server atau dibagi di antara keduanya (lihat Gambar 11-3). Klien yang ditunjukkan pada Gambar 11-3 dapat disebut sebagai klien tebal, atau gemuk, jika berisi sebagian besar logika aplikasi. Praktik saat ini adalah membuat arsitektur client-server menggunakan thin client karena lebih sedikit overhead dan pemeliharaan dalam mendukung aplikasi thin-client. Sebagai contoh, banyak sistem berbasis Web dirancang dengan browser Web melakukan presentasi, dengan hanya logika aplikasi minimal menggunakan bahasa pemrograman seperti Java dan server Web memiliki logika aplikasi, logika akses data, dan penyimpanan data.

Arsitektur client-server memiliki empat manfaat penting. Pertama, mereka terukur. Itu berarti mudah untuk menambah atau mengurangi kemampuan penyimpanan dan pemrosesan server. Jika satu server menjadi kelebihan beban, Anda cukup menambahkan server lain sehingga banyak server yang digunakan untuk menjalankan logika aplikasi, logika akses data, atau penyimpanan data. Biaya untuk meningkatkan jauh lebih bertahap, dan Anda dapat meningkatkan dalam langkah-langkah yang lebih kecil daripada menghabiskan ratusan ribu untuk meningkatkan server mainframe.

Arsitektur client-server dapat mendukung berbagai jenis klien dan server. Dimungkinkan untuk menghubungkan komputer yang menggunakan sistem operasi yang berbeda sehingga pengguna dapat memilih jenis komputer yang mereka sukai (misalnya, menggabungkan komputer Windows dan Apple Macintoshes di jaringan yang sama). Kami

tidak terutama dalam satu vendor, seperti yang sering terjadi pada jaringan berbasis server. Middleware adalah jenis perangkat lunak sistem yang dirancang untuk menerjemahkan antara perangkat lunak vendor yang berbeda. Middleware diinstal pada komputer klien dan komputer server. Perangkat lunak klien berkomunikasi dengan middleware, yang dapat memformat ulang pesan ke dalam bahasa standar yang dapat dipahami oleh middleware yang membantu perangkat lunak server.



Gambar 11-3 Arsitektur Client-Server

Untuk arsitektur thin-client server yang menggunakan standar Internet, sangatlah mudah untuk memisahkan logika presentasi, logika aplikasi, dan logika serta desain akses data dengan jelas sehingga masing-masing agak independen. Misalnya, logika presentasi dapat dirancang dalam HTML atau XML untuk menentukan bagaimana halaman akan muncul di layar (lihat Bab 10). Pernyataan program sederhana digunakan untuk menghubungkan bagian antarmuka ke modul logika aplikasi tertentu yang melakukan berbagai fungsi. File HTML atau XML yang mendefinisikan antarmuka ini dapat diubah tanpa mempengaruhi logika aplikasi. Demikian juga, dimungkinkan untuk mengubah logika aplikasi tanpa mengubah logika presentasi atau data, yang disimpan dalam database dan diakses menggunakan perintah SQL.

Terakhir, karena tidak ada satu pun komputer server yang mendukung semua aplikasi, jaringan umumnya lebih andal. Tidak ada titik pusat kegagalan yang akan menghentikan seluruh jaringan jika gagal, seperti yang ada dalam komputasi berbasis server. Jika salah satu server gagal di lingkungan client-server, jaringan dapat terus berfungsi menggunakan semua server lain (tetapi, tentu saja, aplikasi apa pun yang memerlukan server yang gagal tidak akan berfungsi).

Arsitektur client-server juga memiliki beberapa batasan kritis, yang paling penting adalah kompleksitasnya. Semua aplikasi dalam komputasi client-server memiliki dua bagian, perangkat lunak pada klien dan perangkat lunak pada server. Menulis perangkat lunak ini lebih rumit daripada menulis perangkat lunak all-in-one tradisional yang digunakan dalam arsitektur berbasis server. Memperbarui jaringan dengan versi baru perangkat lunak juga lebih rumit. Dalam arsitektur berbasis server, ada satu tempat di mana perangkat lunak aplikasi disimpan; untuk mengupdate software, kita cukup menggantinya disana. Dengan arsitektur client-server, kita harus memperbarui semua klien dan semua server.

Sebagian besar perdebatan tentang arsitektur berbasis server versus klien-server berpusat pada biaya. Salah satu klaim besar jaringan berbasis server pada 1980-an adalah bahwa mereka menyediakan skala ekonomi. Produsen mainframe besar mengklaim lebih

murah untuk menyediakan layanan komputer pada satu mainframe besar daripada satu set komputer yang lebih kecil. Revolusi komputer pribadi mengubah ini. Sejak tahun 1980-an, biaya komputer pribadi terus menurun, sedangkan kinerjanya meningkat secara signifikan. Saat ini, perangkat keras komputer pribadi lebih dari 1.000 kali lebih murah daripada perangkat keras mainframe untuk jumlah daya komputasi yang sama.

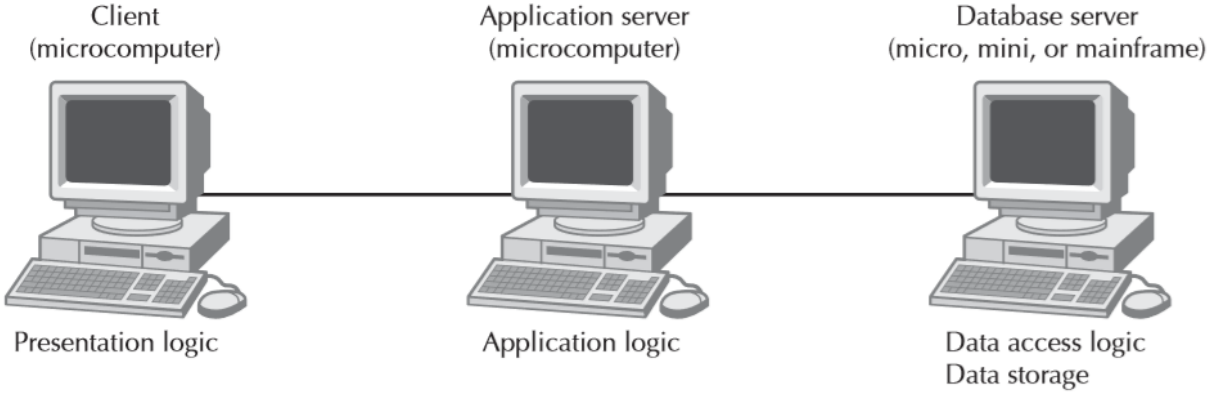
Dengan perbedaan biaya seperti ini, mudah untuk melihat mengapa tiba-tiba terjadi lonjakan komputasi klien-server berbasis komputer mikro. Masalah dengan perbandingan biaya ini adalah bahwa mereka mengabaikan total biaya kepemilikan, yang mencakup faktor-faktor selain biaya perangkat keras dan perangkat lunak yang jelas. Misalnya, banyak perbandingan biaya mengabaikan peningkatan kompleksitas yang terkait dengan pengembangan perangkat lunak aplikasi untuk jaringan client-server. Kebanyakan ahli percaya bahwa biaya empat sampai lima kali lebih banyak untuk mengembangkan dan memelihara perangkat lunak aplikasi untuk komputasi client-server daripada untuk komputasi berbasis server.

Server klien

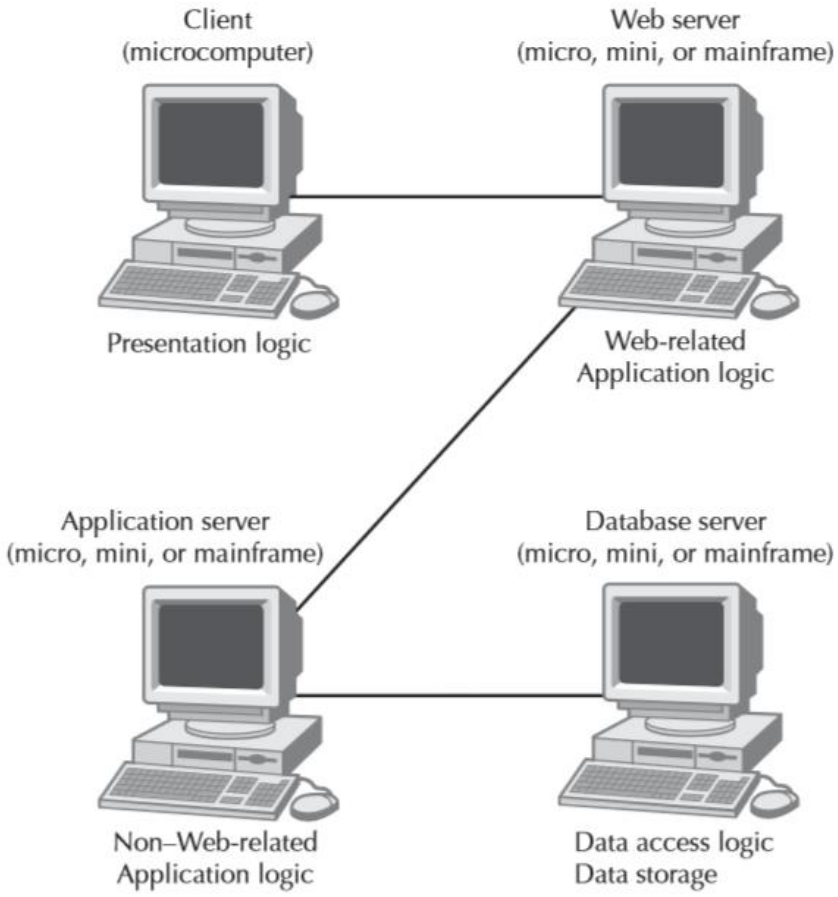
Tingkatan Ada banyak cara logika aplikasi dapat dipartisi antara klien dan server. Contoh pada Gambar 11-3 adalah salah satu yang paling umum. Dalam hal ini, server bertanggung jawab atas data, dan klien bertanggung jawab atas aplikasi dan presentasi. Ini disebut arsitektur dua tingkat karena hanya menggunakan dua set komputer, klien, dan server.

Arsitektur tiga tingkat menggunakan tiga set komputer (lihat Gambar 11-4). Dalam hal ini, perangkat lunak pada komputer klien bertanggung jawab atas logika presentasi, server aplikasi (atau server) bertanggung jawab atas logika aplikasi, dan server database (atau server) terpisah bertanggung jawab atas logika akses data dan penyimpanan data.

Arsitektur n-tier menggunakan lebih dari tiga set komputer. Dalam hal ini, klien bertanggung jawab untuk presentasi, server database bertanggung jawab atas logika akses data dan penyimpanan data, dan logika aplikasi tersebar di dua atau lebih set server yang berbeda Jenis arsitektur ini umum dalam sistem e-commerce saat ini (lihat Gambar 11-5). Komponen pertama adalah browser Web pada komputer klien yang digunakan oleh pengguna untuk mengakses sistem dan memasukkan perintah (logika presentasi). Yang kedua adalah server Web yang merespon permintaan pengguna, baik dengan menyediakan halaman (HTML) dan grafik (logika aplikasi) atau dengan mengirimkan permintaan ke komponen ketiga pada server aplikasi lain yang melakukan berbagai fungsi (logika aplikasi). Komponen keempat adalah database server yang menyimpan semua data (logika akses data dan penyimpanan data). Masing-masing dari keempat komponen ini terpisah, sehingga mudah untuk menyebarkan komponen yang berbeda pada server yang berbeda dan untuk mempartisi logika aplikasi pada dua server yang berbeda.



Gambar 11-4 Arsitektur Tiga-Tiered Client-Server



Gambar 11-5 Arsitektur Klien-Server Empat Tingkat

Keuntungan utama dari arsitektur client-server n-tier dibandingkan dengan arsitektur twotiered (atau arsitektur tiga-tier dengan arsitektur two-tier) adalah memisahkan pemrosesan yang terjadi untuk lebih menyeimbangkan beban pada server yang berbeda; itu lebih terukur. Pada Gambar 11-5, kami memiliki tiga server terpisah, sebuah konfigurasi yang menyediakan daya lebih besar daripada jika kami menggunakan arsitektur dua tingkat dengan hanya satu server. Jika kami menemukan bahwa server aplikasi terlalu banyak dimuat, kami dapat dengan mudah menggantinya dengan server yang lebih kuat atau hanya memasukkan beberapa server aplikasi lagi. Sebaliknya, jika kami menemukan server database kurang digunakan, kami dapat menyimpan data dari aplikasi lain di dalamnya.

Ada dua kelemahan utama dari arsitektur n-tier dibandingkan dengan arsitektur two-tier (atau arsitektur three-tier dengan arsitektur two-tier). Pertama, konfigurasi memberikan beban yang lebih besar pada jaringan. Jika Anda membandingkan Gambar 11-3, 11-4, dan 11-5, Anda akan melihat bahwa model n-tier membutuhkan lebih banyak komunikasi antar server; ini menghasilkan lebih banyak lalu lintas jaringan, jadi Anda memerlukan jaringan berkapasitas lebih tinggi. Juga jauh lebih sulit untuk memprogram dan menguji perangkat lunak dalam arsitektur n-tier daripada di arsitektur dua-tier karena lebih banyak perangkat harus berkomunikasi untuk menyelesaikan transaksi pengguna.

Memilih Arsitektur Fisik

Sebagian besar sistem dibangun untuk menggunakan infrastruktur yang ada dalam organisasi, sehingga seringkali infrastruktur saat ini membatasi pilihan arsitektur. Misalnya, jika sistem baru akan dibangun untuk organisasi yang berpusat pada mainframe, arsitektur

berbasis server mungkin merupakan pilihan terbaik. Faktor lain seperti standar perusahaan, perjanjian lisensi yang ada, dan hubungan produk/vendor juga dapat mengamanatkan arsitektur apa yang perlu dirancang oleh tim proyek. Namun, banyak organisasi sekarang memiliki berbagai infrastruktur yang tersedia atau secara terbuka mencari proyek percontohan untuk menguji arsitektur dan infrastruktur baru, memungkinkan tim proyek untuk memilih arsitektur berdasarkan faktor penting lainnya.

Masing-masing arsitektur komputasi yang baru saja dibahas memiliki kekuatan dan kelemahannya, dan tidak ada arsitektur yang secara inheren lebih baik dari yang lain. Dengan demikian, penting untuk memahami kekuatan dan kelemahan masing-masing arsitektur komputasi dan kapan menggunakannya. Gambar 11-6 menyajikan ringkasan karakteristik penting masing-masing.

Biaya Infrastruktur. Salah satu kekuatan pendorong terkuat untuk arsitektur client-server adalah biaya infrastruktur (perangkat keras, perangkat lunak, dan jaringan yang akan mendukung sistem aplikasi). Sederhananya, komputer pribadi lebih dari 1.000 kali lebih murah daripada mainframe untuk jumlah daya komputasi yang sama. Komputer pribadi di meja kita saat ini memiliki lebih banyak kekuatan pemrosesan, memori, dan ruang hard disk daripada mainframe biasa di masa lalu, dan biaya komputer pribadi adalah sebagian kecil dari biaya mainframe.

Oleh karena itu, biaya arsitektur client-server rendah dibandingkan dengan arsitektur berbasis server yang mengandalkan mainframe. Arsitektur client-server juga cenderung lebih murah daripada arsitektur berbasis client karena mereka menempatkan lebih sedikit beban pada jaringan dan dengan demikian memerlukan lebih sedikit kapasitas jaringan.

Biaya Pembangunan. Biaya pengembangan sistem merupakan faktor penting ketika mempertimbangkan manfaat finansial dari arsitektur client-server. Mengembangkan perangkat lunak aplikasi untuk komputasi client-server sangat kompleks, dan sebagian besar ahli percaya bahwa pengembangan dan pemeliharaan perangkat lunak aplikasi untuk komputasi client-server membutuhkan biaya empat hingga lima kali lebih banyak daripada yang dilakukan untuk komputasi berbasis server. Mengembangkan perangkat lunak aplikasi untuk arsitektur berbasis klien biasanya masih lebih murah, karena ada banyak alat pengembangan GUI untuk komputer sederhana yang berdiri sendiri yang berkomunikasi dengan server database.

Perbedaan biaya mungkin berubah karena lebih banyak perusahaan mendapatkan pengalaman dengan aplikasi client-server, produk client-server baru dikembangkan dan disempurnakan, dan standar client-server matang. Namun, mengingat kompleksitas yang melekat pada perangkat lunak klien-server dan kebutuhan untuk mengoordinasikan interaksi perangkat lunak pada komputer yang berbeda, kemungkinan akan tetap ada perbedaan biaya.

Characteristic	Server-based	Client-based	Client–Server
Cost of infrastructure	Very high	Medium	Low
Cost of development	Medium	Low	High
Ease of development	Low	High	Low to medium
Interface capabilities	Low	High	High
Control and security	High	Low	Medium
Scalability	Low	Medium	High

Gambar 11-6 Karakteristik Arsitektur Komputasi

Kemudahan Pengembangan. Di sebagian besar organisasi saat ini, ada tumpukan besar aplikasi mainframe, sistem yang telah disetujui tetapi kekurangan staf untuk mengimplementasikannya. Backlog ini menandakan kesulitan dalam mengembangkan sistem berbasis server. Alat untuk sistem berbasis mainframe sering kali tidak ramah pengguna dan memerlukan keterampilan yang sangat khusus—keterampilan yang sering tidak dimiliki oleh lulusan baru dan tidak tertarik untuk memperolehnya. Sebaliknya, arsitektur berbasis klien dan klien-server dapat mengandalkan alat pengembangan antarmuka pengguna grafis/*graphical user interface* (GUI) yang intuitif dan mudah digunakan. Pengembangan aplikasi untuk arsitektur ini bisa cepat dan tidak menyakitkan. Sayangnya, aplikasi untuk sistem client-server bisa sangat kompleks karena harus dibangun untuk beberapa lapisan perangkat keras (misalnya, server database, server Web, workstation klien) yang perlu berkomunikasi secara efektif satu sama lain. Tim proyek sering meremehkan upaya yang terlibat dalam menciptakan aplikasi client-server yang aman dan efisien.

Kemampuan Antarmuka. Biasanya, aplikasi berbasis server berisi antarmuka sederhana berbasis karakter. Misalnya, pikirkan tentang sistem reservasi maskapai penerbangan seperti SABRE, yang bisa sangat sulit digunakan kecuali jika operator terlatih dengan baik tentang perintah dan ratusan kode yang digunakan untuk menavigasi sistem. Saat ini, sebagian besar pengguna sistem mengharapkan GUI atau antarmuka berbasis Web yang dapat mereka operasikan menggunakan mouse dan objek grafis. GUI dan alat pengembangan Web biasanya dibuat untuk mendukung aplikasi berbasis klien atau klien-server; jarang lingkungan berbasis server dapat mendukung jenis aplikasi ini.

Kontrol dan Keamanan. Arsitektur berbasis server pada awalnya dikembangkan untuk mengontrol dan mengamankan data, dan jauh lebih mudah untuk dikelola karena semua data disimpan di satu lokasi. Sebaliknya, komputasi client-server membutuhkan tingkat koordinasi yang tinggi di antara banyak komponen, dan peluang untuk lubang keamanan atau masalah kontrol jauh lebih mungkin. Juga, perangkat keras dan perangkat lunak yang digunakan dalam arsitektur client-server masih matang dalam hal keamanan. Ketika sebuah organisasi memiliki sistem yang mutlak harus aman, maka tim proyek mungkin lebih nyaman dengan alternatif berbasis server pada komputer mainframe yang sangat aman dan berorientasi kontrol.

Skalabilitas. Skalabilitas mengacu pada kemampuan untuk menambah atau mengurangi kapasitas infrastruktur komputasi dalam menanggapi perubahan kebutuhan kapasitas. Arsitektur yang paling scalable adalah komputasi client-server karena server dapat ditambahkan ke (atau dihapus dari) arsitektur ketika kebutuhan pemrosesan berubah. Juga, jenis perangkat keras yang digunakan dalam situasi klien-server biasanya dapat ditingkatkan dengan kecepatan yang paling sesuai dengan pertumbuhan aplikasi. Sebaliknya, arsitektur berbasis server mengandalkan terutama pada perangkat keras mainframe yang perlu ditingkatkan dalam jumlah besar, peningkatan yang mahal, dan arsitektur berbasis klien memiliki batas di mana aplikasi tidak dapat tumbuh karena peningkatan penggunaan dan data dapat mengakibatkan peningkatan lalu lintas jaringan ke sejauh mana kinerja tidak dapat diterima.

11.4 KOMPUTASI CLOUD

Komputasi Cloud adalah gagasan memperlakukan TI sebagai utilitas atau komoditas. Pada dasarnya, Komputasi Cloud adalah pendekatan terbaru untuk mendukung komputasi terdistribusi dalam jenis arsitektur client-server (lihat bagian sebelumnya) di mana server "di awan" dan klien di desktop. Cloud dapat berupa pusat data perusahaan perusahaan, pusat data eksternal, atau kombinasi keduanya; namun, lebih dan lebih umumnya dilihat sebagai layanan eksternal, bukan internal. Akibatnya, gagasan multitenancy, di mana vendor cloud memiliki

banyak pelanggan yang menggunakan sumber daya yang sama pada saat yang sama, menjadi masalah nyata bagi vendor cloud dan pelanggan cloud. Komputasi Cloud mungkin menjadi pendorong terbesar untuk outsourcing TI (lihat Bab 7).

Ada tiga klasifikasi awan yang berbeda: privat, publik, dan hybrid. Awan pribadi hanya tersedia untuk karyawan perusahaan, awan publik tersedia untuk masyarakat umum, dan awan hibrida menggabungkan ide-ide awan pribadi dan publik untuk membentuk awan tunggal. Dalam beberapa hal, semua situs e-commerce dapat berjalan di lingkungan cloud hybrid di mana bagian transaksi penjualan pelanggan dari sistem harus bersifat publik sementara semua aspek lainnya bersifat pribadi.

Pada dasarnya, Komputasi Cloud adalah teknologi payung yang mencakup ide-ide virtualisasi, arsitektur berorientasi layanan, dan komputasi grid. Ide virtualisasi bukanlah hal baru. Virtualisasi adalah gagasan untuk memperlakukan sumber daya komputasi apa pun, di mana pun ia berada, seolah-olah "di" mesin klien. Ide ini berkembang dari memori virtual. Memori virtual awalnya dikembangkan pada tahun 1960-an. Memori virtual memungkinkan pengguna / programmer untuk bertindak seolah-olah jumlah memori utama di komputer tidak terbatas. Ini dilakukan dengan menukar "halaman" memori utama ke disk ketika konten halaman tidak digunakan dan dengan menukar halaman dari disk kembali ke memori utama saat dibutuhkan. Sebelum memori virtual dibuat, programmer harus menulis kode untuk melakukan fungsi paging untuk setiap aplikasi. Virtualisasi hanyalah peningkatan ide ini ke semua sumber daya komputasi, bukan hanya memori utama. Ini termasuk memperlakukan komputer mainframe seolah-olah itu adalah satu set server virtual, yang masing-masing dapat menjalankan sistem operasi dan/atau aplikasi yang berbeda.

Layanan web pada dasarnya mendukung koneksi antara layanan yang berbeda untuk membentuk arsitektur berorientasi layanan.4 Pada dasarnya, layanan adalah bagian dari perangkat lunak yang mendukung beberapa aspek dari proses bisnis. Sebuah layanan dapat menjadi implementasi bagian dari proses bisnis, dapat menjadi implementasi dari seluruh proses bisnis, atau dapat menjadi dukungan persistensi objek untuk lapisan manajemen data (lihat Bab 9). Layanan ini dapat berupa internal atau eksternal perusahaan. Layanan dapat digabungkan untuk mendukung proses bisnis. Arsitektur berorientasi layanan memungkinkan proses bisnis didukung oleh layanan "plug and play" bersama-sama secara statis dan/atau dinamis. Beberapa layanan yang dapat dipasang dan dimainkan dapat dibeli langsung, atau dapat ditagihkan ke perusahaan berdasarkan penggunaannya, semacam model bayar sesuai pemakaian.

Komputasi grid cenderung menjadi teknologi perangkat keras yang mendasari yang mendukung cloud. Grid adalah sekumpulan komputer jaringan yang sangat besar yang cenderung tersebar secara geografis. Misalnya, kisi yang mendukung aplikasi CRM SalesForce.com berisi sekitar 1.000 komputer. Komputer tidak harus dari jenis yang sama. Misalnya, mereka dapat menjadi campuran server Linux dan mainframe. Dengan komputasi grid, perusahaan memiliki kemampuan untuk menambah dan menghapus komputer untuk mendukung proses bisnis berdasarkan tingkat aktivitas saat ini yang terjadi dalam proses bisnis tertentu. Ini memberikan sejumlah besar fleksibilitas dalam mengkonfigurasi arsitektur fisik yang mendasari yang mendukung proses bisnis.

Menggabungkan virtualisasi, arsitektur berorientasi layanan, dan komputasi grid adalah semua kehebohan yang berkaitan dengan komputasi awan. Komputasi Cloud sangat elastis dan skalabel, mendukung pendekatan berbasis permintaan untuk penyediaan dan deprovisioning sumber daya, dan mendukung model penagihan yang hanya mengenakan biaya untuk sumber daya yang digunakan. Dari perspektif bisnis, Komputasi Cloud mendukung gagasan bahwa TI menjadi komoditas.

Cloud dapat berisi infrastruktur TI perusahaan, platform TI, dan perangkat lunak. Infrastruktur sebagai Layanan/ *Infrastructure as a Service* (IaaS) mengacu pada cloud yang menyediakan perangkat keras komputasi kepada perusahaan sebagai layanan jarak jauh. Perangkat keras biasanya mencakup perangkat keras komputasi yang mendukung server aplikasi, jaringan, dan penyimpanan data. Layanan EC2 (aws.amazon.com/ec2/) Amazon adalah contoh yang bagus untuk ini. Dengan *Platform as a Service* (PaaS), vendor cloud tidak hanya menyediakan dukungan perangkat keras kepada pelanggan, tetapi juga menyediakan solusi berbasis paket, layanan berbeda yang dapat digabungkan untuk membuat solusi, atau alat pengembangan yang diperlukan untuk pelanggan. buat solusi khusus di cloud vendor PaaS. SalesForce.com adalah contoh yang baik dari vendor yang menyediakan solusi berbasis paket, SimpleDB Amazon dan Simple Query Service adalah contoh dari berbagai layanan yang didukung, dan Google App Engine adalah contoh vendor cloud yang menyediakan alat pengembangan yang baik. Seperti kebanyakan hal di TI, *Software as a Service* (SaaS) bukanlah ide baru. SaaS telah ada selama lebih dari tiga puluh tahun. Pada 1970-an, ada banyak "biro layanan" yang mendukung pembagian waktu perangkat keras dan perangkat lunak ke banyak pelanggan yang berbeda; yaitu, mereka mendukung multitenancy. Misalnya, ADP telah mendukung fungsi penggajian untuk banyak perusahaan untuk waktu yang sangat lama. Saat ini, sistem CRM SalesForce.com adalah contoh yang baik dari solusi berbasis cloud SaaS.

Namun, Komputasi Cloud harus mengatasi kendala tertentu sebelum menjadi pendekatan utama untuk menyediakan lapisan arsitektur fisik. Kendala pertama adalah tingkat kinerja cloud yang beragam. Satu masalah adalah apakah vendor memiliki sumber daya untuk menyediakan "kekuatan" yang cukup bagi perusahaan selama beban puncak. Masalahnya di sini adalah bahwa vendor cloud biasa mendukung banyak perusahaan yang berbeda. Jika vendor tidak memiliki sumber daya komputasi yang cukup untuk menangani semua beban puncak perusahaan pada saat yang sama, maka harus ada beberapa degradasi dari beberapa atau semua dukungan perusahaan. Ini terutama merupakan hasil dari ketidakpastian persyaratan kinerja keseluruhan dengan I/O disk dan lalu lintas jaringan. Mengingat multitenancy khas perangkat keras vendor cloud, kemacetan dengan disk akan terjadi. Namun, mengingat ketergantungan pada jaringan, kecepatan transfer data sangat penting. Dalam contoh yang mencerahkan, Armbrust dan rekan menunjukkan bahwa ketika berurusan dengan volume data yang besar, lebih cepat untuk mentransfer data menggunakan pengiriman semalam. Dalam contoh mereka, mereka menunjukkan bahwa jika Anda mentransfer 10 terabyte data dengan kecepatan transfer rata-rata 20 Mbits/dtk, maka dibutuhkan lebih dari 45 hari untuk menyelesaikan transfer. Jika Anda mengirimkan data dalam semalam, Anda akan secara efektif menggunakan kecepatan transfer 1500 Mbits/dtk.

Hambatan kedua berkaitan dengan tingkat ketergantungan yang dimiliki perusahaan pelanggan pada vendor cloud. Perusahaan bergantung pada vendor cloud berdasarkan jenis layanan yang mereka gunakan (IaaS, PaaS, dan SaaS), tingkat ketersediaan layanan yang sebenarnya, dan potensi penguncian data. Saat ini, sebagian besar API vendor cloud untuk penyimpanan adalah milik. Akibatnya, data pelanggan menjadi "terutama" ke penyimpanan vendor cloud tertentu. Ini juga berlaku untuk sebagian besar API layanan yang sebenarnya. Akibatnya, pelanggan mendapati diri mereka berharap bahwa vendor cloud akan setara dengan diktator baik hati yang akan bertindak demi kepentingan pelanggan; jika tidak, tingkat layanan aktual yang diberikan dapat terganggu. Mengingat potensi penguncian data dan/atau layanan, pelanggan harus memperhatikan kelayakan vendor cloud. Jika vendor gulung tikar, pelanggan bisa mengikutinya dengan sangat cepat. Jika vendor cloud juga telah melakukan outsourcing ke vendor cloud lain, seperti ke perusahaan disk farm, maka mereka dapat menemukan diri mereka dalam situasi yang sama. Hal ini dapat menyebabkan efek cascading dari kegagalan bisnis. Akibatnya, ketika sebuah perusahaan mempertimbangkan untuk

mengalihdayakan area TI ke cloud, perusahaan harus lebih memahami total risiko yang terlibat.

Kendala utama ketiga untuk adopsi cloud adalah tingkat keamanan yang dirasakan yang tersedia di cloud. Perusahaan tidak hanya harus mengkhawatirkan keamanan dari luar, tetapi karena multitenancy, perusahaan harus secara serius mempertimbangkan potensi serangan dari dalam cloud dari pengguna cloud lainnya. Dari perspektif ketersediaan layanan, serangan penolakan layanan terhadap penyewa lain di dalam cloud dapat menyebabkan penurunan kinerja sistem perusahaan. Akhirnya, perusahaan harus mempertimbangkan untuk melindungi dirinya dari vendor cloud. Vendor cloud hanya bertanggung jawab atas keamanan fisik dan firewall. Semua keamanan tingkat aplikasi cenderung menjadi tanggung jawab pelanggan cloud. Jelas, keamanan di cloud adalah upaya yang sangat kompleks. Mengingat persyaratan kerahasiaan dan auditabilitas *Sarbanes-Oxley* (SOX) dan *Health and Human Services Health Insurance Portability and Accountability Act* (HIPAA), keamanan di cloud menjadi perhatian utama ketika perusahaan mempertimbangkan untuk memindahkan data rahasianya, termasuk e -mail, ke awan. Dalam banyak hal, saat menggunakan cloud, sebuah perusahaan hanya mengambil lompatan keyakinan bahwa cloud itu aman.

11.5 KOMPUTASI DI MANA-MANA DAN INTERNET OF THING

Seringkali, komputasi di mana-mana dan *Internet of Things* (IoT) adalah awal dari realisasi semua mimpi (atau mimpi buruk) penulis fiksi ilmiah. Ini berkisar dari pandangan dystopian yang digambarkan dalam film Blade Runner dan Terminator hingga unit Precrime dari Minority Report dan akhirnya ke masa depan yang sangat optimis yang digambarkan dalam kartun The Jetsons tahun 1960-an. Pada dasarnya, komputasi di mana-mana adalah gagasan bahwa komputasi terjadi di mana-mana dan dalam segala hal. Dengan komputasi di mana-mana, komputasi menjadi begitu mendarah daging ke dalam hal-hal sehari-hari sehingga komputasi secara efektif menghilang ke latar belakang. Dengan kata lain, komputasi menjadi begitu dalam mengakar pada hal-hal sehari-hari sehingga hal-hal itu sendiri tampak menjadi magis. IoT adalah gagasan bahwa, selain hal-hal yang memiliki beberapa bentuk kapasitas komputasi yang terpasang di dalamnya, hal-hal sehari-hari menjadi terhubung melalui Internet. Jadi, selain memiliki beberapa bentuk kapasitas komputasi, hal-hal sehari-hari dapat berkomunikasi satu sama lain. Ini meningkatkan pentingnya memahami komputasi seluler, media sosial, dan Komputasi Cloud lebih jauh. Jelas, peluang (atau jebakan) yang diberikan ini mungkin tidak terbatas.

Saat ini, ada dua pendekatan utama untuk mendukung komputasi di mana-mana: perangkat komputasi umum dan perangkat komputasi khusus. Perangkat komputasi umum termasuk perangkat seperti smartphone dan tablet. Perangkat ini dapat dimuat dengan banyak aplikasi berbeda yang menyediakan semua jenis dukungan komputasi dan komunikasi. Misalnya, ponsel cerdas Anda dapat digunakan sebagai GPS, pembaca e-book, pemutar musik atau video, konsol game, antarmuka WWW, kamera, perekam “tape”, penasihat restoran, dll. Dengan kata lain , jika ada aplikasi untuk itu, Anda dapat memuatnya di ponsel cerdas Anda untuk memberi Anda kemampuan itu. Smartphone saat ini pada dasarnya adalah komputer umum yang mendukung komunikasi suara, yaitu, juga telepon. Dan, seperti komputer untuk keperluan umum, ponsel cerdas biasanya mengharuskan Anda untuk mengaktifkan aplikasi sebelum dapat melakukan apa pun untuk Anda. Meskipun sangat mengesankan untuk memiliki jumlah kemampuan komputasi di ujung jari Anda, itu hanya mendukung impian komputasi di mana-mana dengan cara yang sangat terbatas. Pada dasarnya, dari perspektif pengembangan sistem informasi, ini bukanlah hal baru; tidak ada bedanya dengan memiliki komputer yang terhubung ke Internet. Oleh karena itu, mengembangkan aplikasi untuk perangkat ini harus mengikuti pendekatan pengembangan dasar yang sama yang digunakan di seluruh buku ini.

Pendekatan kedua, memiliki perangkat komputasi khusus, sangat membantu mewujudkan impian komputasi di mana-mana. Dengan pendekatan ini, kami memiliki apa yang disebut objek terpesona yang dapat berinteraksi satu sama lain. Objek terpesona adalah objek sehari-hari yang memiliki prosesor yang sangat khusus tertanam di dalamnya yang menambah objek sedemikian rupa sehingga objek tersebut tampak ajaib. Misalnya, payung yang, karena kemungkinan besar hujan, memberi tahu Anda bahwa Anda harus membawanya hari ini, atau dompet yang memberi tahu Anda bahwa Anda telah mencapai batas anggaran bulanan atau bahwa akun Anda baru saja menerima menyetorkan. Dalam kasus payung, payung terhubung ke AccuWeather. Jika ramalan hujan, payung mengaktifkan satu set LED di pegangan yang memberi tahu Anda bahwa Anda harus membawanya saat Anda pergi. Dalam kasus dompet, saat Anda menghabiskan anggaran bulanan Anda, dompet menjadi lebih sulit untuk dibuka, atau jika Anda menerima setoran ke akun Anda, dompet "menggembung" untuk memberi tahu Anda bahwa dompet Anda lebih gemuk, yaitu , Anda memiliki lebih banyak uang tunai yang tersedia.

Pendekatan pengembangan sistem informasi umum yang digunakan dalam buku ini berlaku untuk pengembangan objek terpesona. Namun, mengingat bahwa objek terpesona, menurut definisi, adalah benda sehari-hari yang disempurnakan, masalah tambahan juga harus ditangani. Isu-isu ini termasuk seperangkat prinsip desain yang unik, satu set karakteristik, dan satu set tingkat pesona.

McEwen dan Cassimally mengidentifikasi seperangkat prinsip desain unik yang perlu dipertimbangkan saat mengembangkan objek terpesona. Pertama, objek terpesona harus di latar belakang hanya memberikan pesannya untuk Anda terima di waktu luang Anda, bukan "di wajah Anda." Ini berbeda dengan kebanyakan aplikasi saat ini. Biasanya, aplikasi akan memberi tahu Anda tentang beberapa topik di waktu luang mereka dengan mengganggu Anda. Kedua, sihir adalah metafora yang berguna bagi orang untuk mengadopsi objek terpesona. Payung yang disebutkan sebelumnya adalah contoh yang baik dari prinsip ini. Payung hanya duduk di dekat pintu memberi tahu Anda apakah itu ingin dibawa atau tidak. Ketiga adalah seluruh masalah privasi. Dengan semua objek terpesona ini "berbagi" data tentang Anda, semua masalah yang terkait dengan "Kakak" Orwell menjadi fokus. Bagaimana Anda akan merahasiakan sesuatu dan, mungkin yang lebih penting, siapa yang sebenarnya memiliki data yang dikumpulkan? Namun, masalah ini tidak unik untuk objek terpesona. Ini juga berlaku untuk ponsel cerdas dan aplikasinya. Keempat, kita perlu mempertimbangkan bagaimana "menggabungkan" satu set objek terpesona yang terhubung secara longgar untuk mendukung tujuan yang lebih besar. Faktanya, Brynjolfsson dan McAfee menyarankan bahwa jenis inovasi rekombinan ini dapat memberikan dasar bagi jenis ekonomi baru yang akan meningkatkan kemajuan dan kemakmuran. Kelima, gagasan tentang keterjangkauan menjadi semakin penting. Agar objek terpesona diadopsi, itu harus sangat mudah digunakan. Objek itu sendiri harus menyiratkan bagaimana menggunakannya. Payung, misalnya, hanya memberi tahu Anda bahwa Anda harus membawanya dengan menarik perhatian Anda padanya.

Rose menyediakan serangkaian karakteristik yang harus dimiliki oleh objek terpesona jika kita ingin mengadopsinya. Pertama, mereka harus bisa dilirik. Payung, sekali lagi, adalah contoh yang bagus. Anda tidak perlu melakukan apa pun selain melirik payung untuk mengetahui apakah Anda harus membawanya atau tidak. Kedua, objek yang disihir harus dapat diberi isyarat. Hal ini terkait dengan keterjangkauan ide. Itu harus jelas secara intuitif tentang cara menggunakan objek terpesona. Misalnya, The A.T. Cross Company menjual buku catatan dan kombinasi pena (CrossPadTM) yang dapat Anda gunakan untuk membuat catatan. Keterjangkauan produk ini adalah kenyataan bahwa Anda hanya menggunakan pena khusus untuk menulis catatan Anda di atas kertas yang terdapat di buku catatan. Bagian yang menarik adalah fakta bahwa produk tersebut juga memiliki pemancar radio yang terpasang di

pena yang memungkinkannya menyimpan catatan Anda dalam bentuk elektronik yang dapat diunggah ke komputer Anda nanti. Ketiga, objek yang disihir harus terjangkau. Dalam hal ini, mengingat turunnya biaya perangkat keras komputasi, jika objeknya tidak begitu terjangkau pada awalnya, itu harus cukup cepat. Keempat, objek harus dapat dipakai. Nike FuelBand™ adalah contoh sempurna. Anda cukup memakainya seperti jam tangan. Kelima, objek terpesona harus tidak bisa dihancurkan. Jelas, yang ini hanya akan benar karena terkait dengan objek yang mendasarinya. Misalnya, payung ajaib tidak bisa dihancurkan seperti payung biasa, tetapi tidak bisa dihancurkan seperti hal-hal lain di dunia nyata. Keenam, objek yang disihir harus mampu melakukan tugasnya dengan interaksi minimal dengan pengguna. Misalnya, Anda cukup memakai Nike FuelBand™ dan memasangnya di malam hari ke komputer Anda, dan itu akan memperbarui sendiri, mengisi ulang sendiri, memperbarui profil Anda, dan menunggu Anda untuk memakainya di pagi hari. Ketujuh, objek yang disihir harus dapat dicintai. Maksud kami, mereka harus mudah diantropomorfisasi. Kita harus menikmati menggunakannya, dan kita harus melewatkannya ketika tidak. Jelas, Anda harus menyadari bahwa ada trade-off s di antara beberapa karakteristik dan, dengan demikian, tidak semua objek akan memiliki semuanya. Namun, sebagai seorang desainer, objek terpesona Anda harus memiliki sebanyak mungkin.

Rose juga menyarankan serangkaian level (atau langkah) pesona yang harus diperhatikan oleh perancang objek terpesona. Untuk tingkat pertama, ia menyarankan bahwa objek yang disihir harus ditambah objek sehari-hari yang terhubung ke jaringan. Ini memungkinkan mereka untuk mengirim dan menerima data yang dapat digunakan oleh objek terpesona lainnya atau sistem lain. Mengingat jumlah data yang harus dikumpulkan tentang diri kita sendiri dan orang lain untuk saat ini dan di masa depan, tingkat kedua untuk objek terpesona adalah untuk dapat dipersonalisasi sedemikian rupa sehingga mereka dapat berinteraksi dengan kita dengan cara yang disesuaikan. Saat ini, sebagian kecil Amazon dan Netflix sudah melakukan ini, misalnya, dengan daftar rekomendasi yang mereka buat untuk Anda. Rekomendasi mereka didasarkan pada interaksi Anda sebelumnya dengan mereka dan mencocokkan interaksi tersebut dengan interaksi orang lain. Potensi jenis kegiatan ini dalam sistem kesehatan sangat besar. Tingkat ketiga adalah tempat objek terpesona kita berinteraksi dengan jejaring sosial kita untuk secara otomatis memberi tahu rekan kerja kita, atau bagian khusus dari mereka, tentang aktivitas kita dengan objek terpesona. Sekali lagi ini bisa sangat berguna dalam sistem kesehatan di mana objek menginformasikan sistem dokter kita atau kelompok pendukung kesehatan kita tentang jenis kegiatan positif (atau negatif) tertentu. Tingkat keempat mengadaptasi ide game ke objek terpesona kita, yaitu, gamifikasi. FuelBand™ Nike adalah contoh sempurna dalam menggunakan gamification untuk membuat pengguna tetap termotivasi secara intrinsik untuk mencapai tujuan individunya. Tingkat terakhir adalah bahwa desainer objek terpesona akan meningkatkan adopsi mereka jika objek dapat menjadi bagian dari sebuah cerita; Rose menyebut ini sebagai story-ification. Melalui penggunaan cerita, pengguna dapat lebih mudah memahami tujuan dan utilitas yang disediakan oleh objek terpesona, sehingga meningkatkan kemungkinan "ikatan" pengguna dengan objek terpesona.

Mengingat potensi komputasi di mana-mana dan IoT, Anda harus mulai mempertimbangkan kemungkinan aplikasi yang mungkin mendapat manfaat darinya. Misalnya, hari ini, melalui penggunaan RFID dan GPS, dimungkinkan untuk mengetahui lokasi setiap item inventaris perusahaan. Meskipun dapat dikatakan bahwa item inventaris dengan tag RFID yang memiliki kemampuan GPS bukanlah objek yang disihir, itu berguna. Dan, meskipun biaya jenis augmentasi ini turun, dapat dikatakan lebih lanjut bahwa Anda mungkin tidak ingin menandai setiap item. Namun, sebelum penglihatan objek terpesona dapat menjadi kenyataan, dua kemungkinan masalah teknis perlu ditangani. Pertama, dengan rangkaian jaringan komunikasi saat ini, dapatkah jaringan saat ini menangani volume

komunikasi tambahan yang diperlukan? Apakah jaringan Internet, ponsel, dan WiFi memiliki bandwidth yang cukup untuk mendukung semua hal tambahan ini? Ketika Anda mulai menghubungkan semuanya ke Internet, diragukan bahwa kapasitasnya ada di sana. Kedua, apakah masuk akal untuk mengharapkan perangkat tujuan khusus sederhana yang tertanam dalam objek terpesona untuk menangani kompleksitas protokol komunikasi yang diperlukan dari jaringan yang ada? Untuk mengatasi masalah ini dapat berarti bahwa arsitektur fisik baru mungkin diperlukan.

11.6 GREEN IT[36]

Mengingat semua kekuatan komputasi yang digunakan untuk memecahkan masalah bisnis saat ini, Green IT menjadi penting. Green IT adalah istilah luas yang mencakup hampir semua hal yang membantu mengurangi dampak lingkungan dari TI. Beberapa topik yang diusung adalah e-waste, penghijauan data center, dan mimpi kantor tanpa kertas.

Pertama, saat membuang perangkat elektronik lama, harus berhati-hati. Komputer lama mengandung bahan yang sangat beracun, termasuk timbal, PCB, merkuri, dan kadmium. Salah satu masalah utama TI Hijau adalah bagaimana membuang limbah elektronik ini. Salah satu tren yang paling mengganggu dalam menangani limbah elektronik adalah pengiriman limbah elektronik dari negara maju ke negara berkembang di mana standar lingkungan hampir tidak ada. Karena teknik "daur ulang halaman belakang" yang digunakan di lokasi ini, bahan beracun yang terkandung dalam limbah elektronik muncul di tanah, air, dan udara. Alternatif untuk sekadar membuang komputer lama ke tempat sampah termasuk memperpanjang siklus penggantian mesin dengan mengubah mesin dari mesin berbasis Windows ke mesin berbasis Linux. Linux membutuhkan lebih sedikit "tenaga kuda" untuk dijalankan daripada Windows. Oleh karena itu, untuk aplikasi tertentu, desktop berbasis Linux lebih dari cukup untuk mengimplementasikan bagian dari lapisan arsitektur fisik.

Kedua, pusat data besar menggunakan listrik sebanyak kota kecil dalam sehari. Akibatnya, mengingat tingkat konsumsi daya ini, menciptakan pusat data hijau di masa depan akan menjadi sangat penting. Ada serangkaian cara untuk membuat pusat data hijau. Salah satu caranya adalah dengan sangat memperhatikan lokasi pusat data. Menempatkan data center di bawah naungan gunung atau gedung tinggi akan mengurangi biaya energi yang dibutuhkan. Misalnya, HP menempatkan salah satu pusat data barunya di timur laut Inggris sehingga dapat didinginkan oleh angin dingin yang bertiup ke pantai dari Laut Utara. Melihat kemungkinan energi alternatif adalah cara lain untuk menangani konsumsi energi. Misalnya, Google telah berkecimpung dalam bisnis membeli ladang angin untuk menghasilkan tenaga bagi pusat datanya, dan HP telah menunjukkan bagaimana pembangkit listrik metana berbasis kotoran sapi dapat dibuat untuk menghasilkan tenaga untuk menjalankan pusat data di negara penghasil susu.

Cara ketiga untuk mempertimbangkan membuat infrastruktur TI Anda lebih hijau adalah dengan mempertimbangkan cloud (lihat bagian sebelumnya). Dengan kemampuan virtualisasi cloud, jumlah server dan desktop berdaya tinggi dapat dikurangi. Namun, Anda perlu melakukan beberapa trade-off antara hambatan pindah ke cloud dan bergerak menuju TI yang lebih hijau. Cara keempat untuk mengatasi permintaan daya untuk infrastruktur TI modern adalah dengan hanya membeli elektronik yang sesuai dengan Energy Star. Cara kelima adalah mendorong karyawan agar mesin mereka "tidur" untuk menghemat energi ketika mesin tidak digunakan selama beberapa waktu.

[36] Caril Baroudi, Jeff rey Hill, Arnold Reinhold, and Jhana Senxian, Green IT for DummiesTM (Hoboken, NJ: Wiley, 2009).

Ide kantor tanpa kertas sudah ada sejak lama. Namun, sampai sekarang, gagasan itu lebih banyak fantasi daripada kenyataan. Saat ini, dengan munculnya tablet multiguna, seperti iPad™ Apple, kantor tanpa kertas menjadi kenyataan. Saat mempertimbangkan cloud dan aplikasi yang tersedia di iPadTM, dimungkinkan tidak hanya untuk membuat kantor tanpa kertas tetapi juga menjadikan kantor tanpa kertas secara efektif menjadi kantor portabel.

11.7 DESAIN INFRASTRUKTUR

Dalam kebanyakan kasus, sistem dibangun untuk organisasi yang sudah memiliki perangkat keras, perangkat lunak, dan infrastruktur komunikasi. Dengan demikian, tim proyek biasanya lebih memperhatikan bagaimana infrastruktur yang ada perlu diubah atau ditingkatkan untuk mendukung persyaratan yang diidentifikasi selama analisis, dibandingkan dengan bagaimana merancang dan membangun infrastruktur dari awal. Koordinasi komponen infrastruktur sangat kompleks, dan membutuhkan profesional teknis yang sangat terampil. Sebagai tim proyek, yang terbaik adalah mengizinkan analis infrastruktur untuk membuat perubahan pada infrastruktur komputasi.

Simpul : ■ Adalah sumber daya komputasi, misalnya, komputer klien, server, jaringan terpisah, atau perangkat jaringan individu. ■ Diberi label dengan namanya. ■ Bisa berisi stereotip untuk memberi label secara khusus jenis simpul yang diwakili, misalnya, perangkat, stasiun kerja klien, server aplikasi, perangkat seluler, dll.	<<stereotype>> Node Name
Artefak: ■ Adalah spesifikasi bagian dari perangkat lunak atau database, misalnya, database atau tabel atau tampilan database, komponen atau lapisan perangkat lunak. ■ Diberi label dengan namanya. ■ Bisa berisi stereotip untuk secara khusus memberi label jenis artefak, misalnya, file sumber, tabel database, file yang dapat dieksekusi, dll.	<<stereotype>> Artifact Name
Simpul dengan artefak yang di-deploy: ■ Menggambarkan artefak yang ditempatkan pada simpul fisik.	<<stereotype>> Node Name <<stereotype>> Artifact Name
Jalur komunikasi: ■ Merupakan pertemuan antara dua simpul. ■ Memungkinkan simpul untuk bertukar pesan. ■ Bisa berisi stereotip yang secara khusus melabeli jenis jalur komunikasi yang diwakili, (misalnya, LAN, Internet, serial, paralel).	<<stereotype>>

Gambar 11-7 Sintaks Diagram Pengembangan

Diagram Penerapan

Diagram penyebaran digunakan untuk mewakili hubungan antara komponen perangkat keras yang digunakan dalam infrastruktur fisik sistem informasi. Misalnya, ketika merancang sistem informasi terdistribusi yang akan menggunakan jaringan area luas, diagram penyebaran dapat digunakan untuk menunjukkan hubungan komunikasi antara node yang

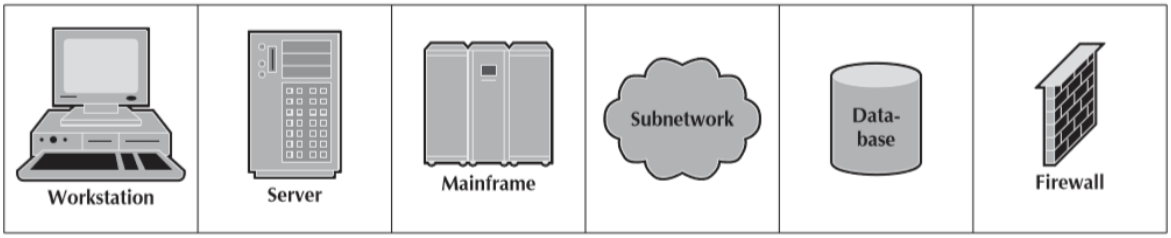
berbeda dalam jaringan. Mereka juga dapat digunakan untuk mewakili komponen perangkat lunak dan bagaimana mereka digunakan di atas arsitektur fisik atau infrastruktur sistem informasi. Dalam hal ini, diagram penyebaran mewakili lingkungan untuk pelaksanaan perangkat lunak.

Unsur-unsur diagram penyebaran termasuk node, artefak, dan jalur komunikasi (lihat Gambar 11-7). Elemen lain juga dapat dimasukkan dalam diagram ini. Dalam kasus kami, kami hanya menyertakan tiga elemen utama dan elemen yang menggambarkan artefak yang dikerahkan ke sebuah node.

Sebuah node mewakili setiap bagian dari perangkat keras yang perlu dimasukkan dalam model desain lapisan arsitektur fisik. Misalnya, node biasanya mencakup komputer klien, server, jaringan terpisah, atau perangkat jaringan individual. Biasanya, sebuah node diberi label dengan namanya dan, mungkin, dengan stereotip. Stereotip dimodelkan sebagai item teks yang dikelilingi oleh simbol "<< >>". Stereotip mewakili jenis simpul yang diwakili pada diagram. Misalnya, stereotip tipikal termasuk perangkat, perangkat seluler, server database, server Web, dan server aplikasi. Ada kalanya notasi node harus diperluas untuk mengkomunikasikan desain lapisan arsitektur fisik dengan lebih baik. Gambar 11-8 mencakup satu set simbol node jaringan tipikal yang dapat digunakan sebagai pengganti notasi standar.

Artefak mewakili bagian dari sistem informasi yang akan digunakan ke arsitektur fisik (lihat Gambar 11-7). Biasanya, artefak mewakili komponen perangkat lunak, subsistem, tabel database, seluruh database, atau lapisan (manajemen data, interaksi manusia-komputer, atau domain masalah). Artefak, seperti node, dapat diberi label dengan nama dan stereotip. Stereotip untuk artefak termasuk file sumber, tabel database, dan file yang dapat dieksekusi.

Jalur komunikasi mewakili hubungan komunikasi antara node arsitektur fisik (lihat Gambar 11-7). Jalur komunikasi distereotipkan berdasarkan jenis tautan komunikasi yang diwakilinya (mis., LAN, Internet, serial, paralel, atau USB) atau protokol yang didukung oleh tautan (mis., TCP/IP).



Gambar 11-8 Extended Node Syntax untuk Development Diagram

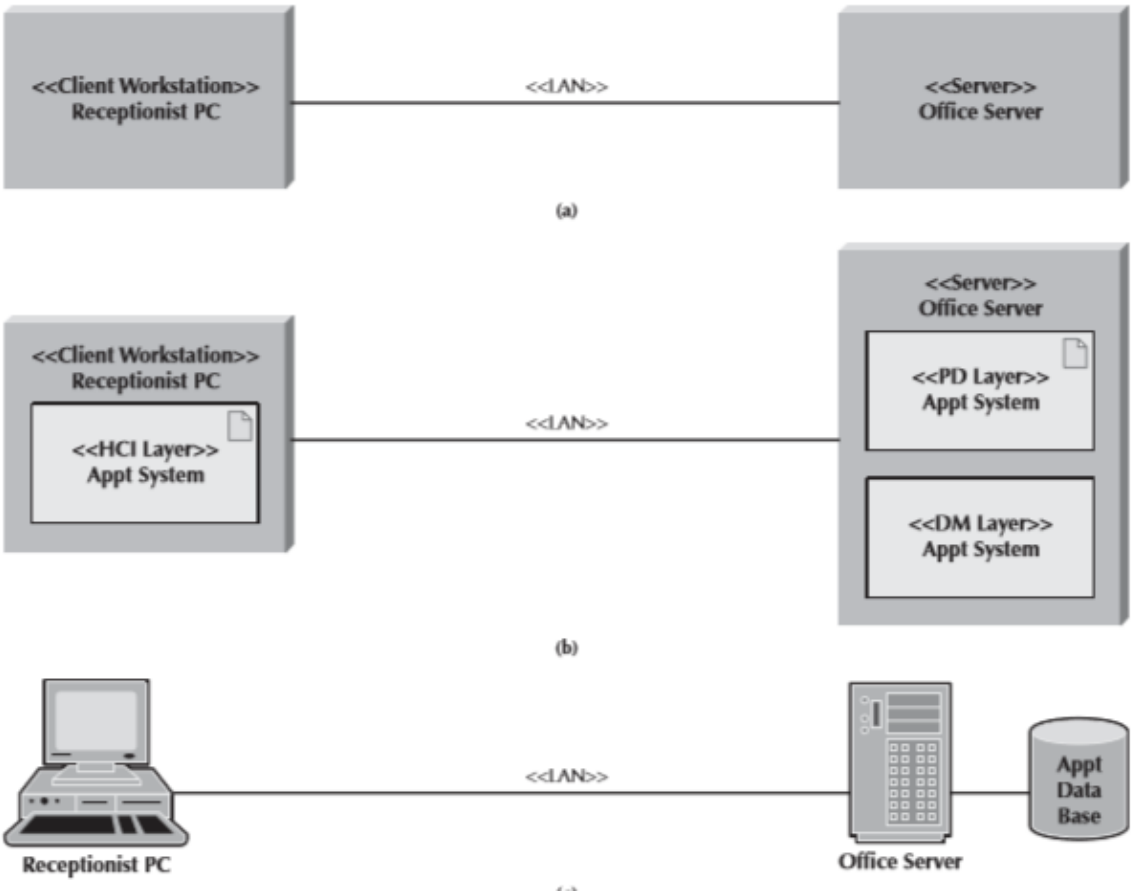
Gambar 11-9 menggambarkan tiga versi berbeda dari diagram penerapan. Versi a hanya menggunakan notasi standar dasar. Versi b memperkenalkan ide untuk menyebarkan artefak ke sebuah node (lihat Gambar 11-7). Dalam hal ini, artefak mewakili lapisan berbeda dari sistem penunjukan yang dijelaskan dalam bab-bab sebelumnya. Versi c menggunakan notasi yang diperluas untuk mewakili arsitektur yang sama. Seperti yang Anda lihat, ketiga versi memiliki kekuatan dan kelemahannya masing-masing. Saat membandingkan versi a dan versi b, pengguna dapat memperoleh lebih banyak informasi dari versi b dengan sedikit usaha tambahan. Namun, ketika membandingkan versi a ke versi c, notasi node yang diperluas memungkinkan pengguna untuk dengan cepat memahami persyaratan perangkat keras dari arsitektur. Ketika membandingkan versi b ke versi c, versi b mendukung distribusi perangkat lunak secara eksplisit tetapi memaksa pengguna untuk mengandalkan stereotip untuk memahami perangkat keras yang diperlukan, sedangkan versi menghilangkan informasi

distribusi perangkat lunak sepenuhnya. Kami menyarankan Anda menggunakan kombinasi simbol untuk menggambarkan arsitektur fisik dengan baik ke komunitas pengguna.

Model Jaringan

Model jaringan adalah diagram yang menunjukkan komponen utama dari sistem informasi (misalnya, server, jalur komunikasi, jaringan) dan lokasi geografis mereka di seluruh organisasi. Tidak ada satu cara untuk menggambarkan model jaringan, dan dalam pengalaman kami, analis membuat standar dan simbol mereka sendiri, menggunakan aplikasi presentasi (misalnya, PowerPoint) atau alat diagram (misalnya, Visio). Dalam teks ini, kami menggunakan diagram penyebaran UML.

Tujuan dari model jaringan ada dua: untuk menyampaikan kompleksitas sistem dan untuk menunjukkan bagaimana komponen perangkat lunak sistem akan cocok bersama. Diagram juga membantu tim proyek mengembangkan spesifikasi perangkat keras dan perangkat lunak yang akan dijelaskan nanti dalam bab ini.

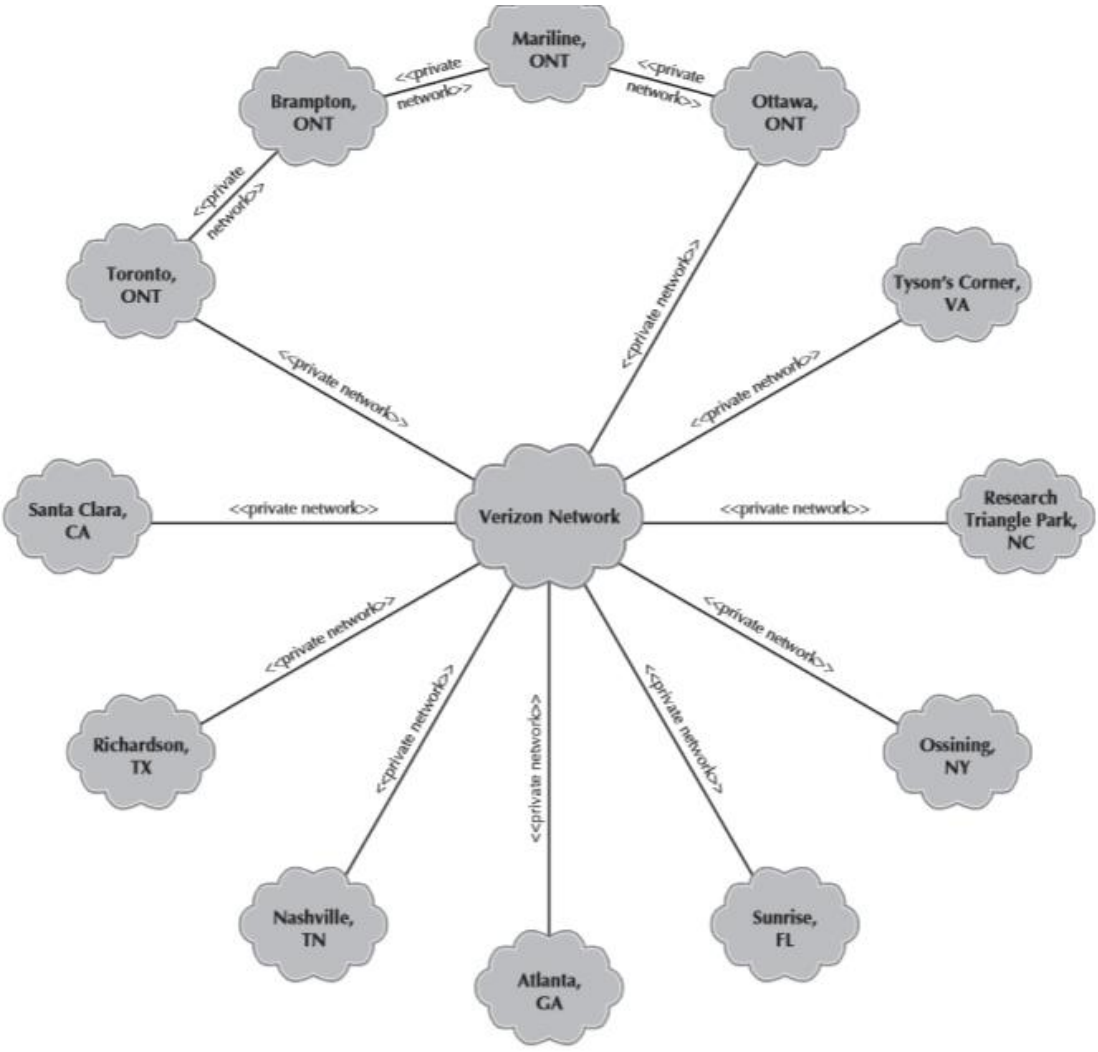


Gambar 11-9 Tiga Versi Diagram Penerapan Sistem Penunjukan

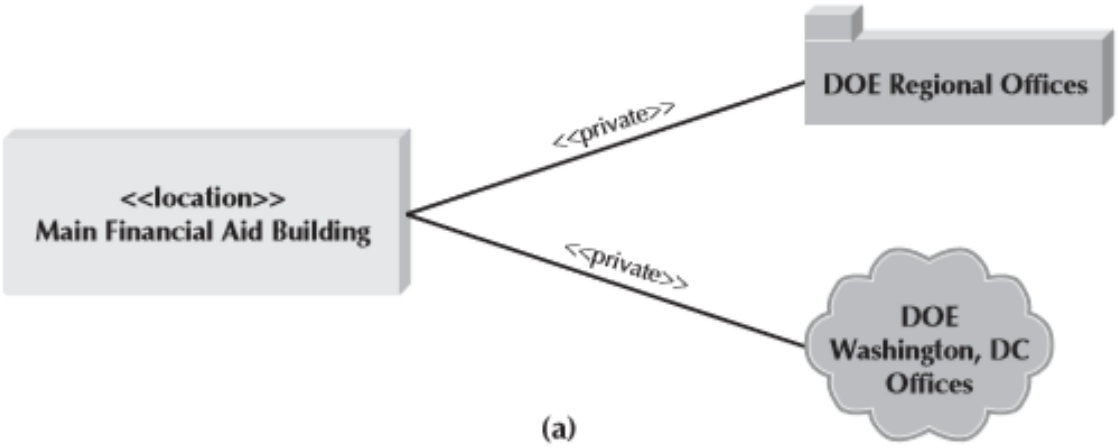
Diagram tingkat tinggi ini memiliki beberapa tujuan. Pertama, menunjukkan lokasi komponen yang dibutuhkan untuk mendukung aplikasi; oleh karena itu, tim proyek dapat memperoleh pemahaman yang baik tentang cakupan geografis sistem baru dan seberapa kompleks dan mahal infrastruktur komunikasi yang akan didukung. (Misalnya, aplikasi yang mendukung satu situs mungkin akan memiliki biaya komunikasi yang lebih sedikit dibandingkan dengan aplikasi yang lebih kompleks yang akan digunakan bersama di seluruh dunia.) Diagram juga menunjukkan komponen eksternal sistem (misalnya, sistem pelanggan , sistem pemasok), yang dapat berdampak pada keamanan atau kebutuhan global (dibahas nanti dalam bab ini).

Langkah kedua dari model jaringan adalah membuat diagram jaringan tingkat rendah untuk setiap lokasi yang ditunjukkan pada diagram tingkat atas. Pertama, perangkat keras digambar pada model dengan cara yang menggambarkan bagaimana perangkat keras untuk sistem baru akan ditempatkan di seluruh lokasi. Biasanya membantu menggunakan simbol yang menyerupai perangkat keras yang akan digunakan. Jumlah detail yang disertakan pada model jaringan tergantung pada kebutuhan proyek. Beberapa model jaringan tingkat rendah berisi deskripsi teks di bawah setiap komponen perangkat keras yang menjelaskan secara rinci konfigurasi perangkat keras yang diusulkan dan kebutuhan pemrosesan; yang lain hanya mencakup jumlah pengguna yang terkait dengan bagian diagram yang berbeda.

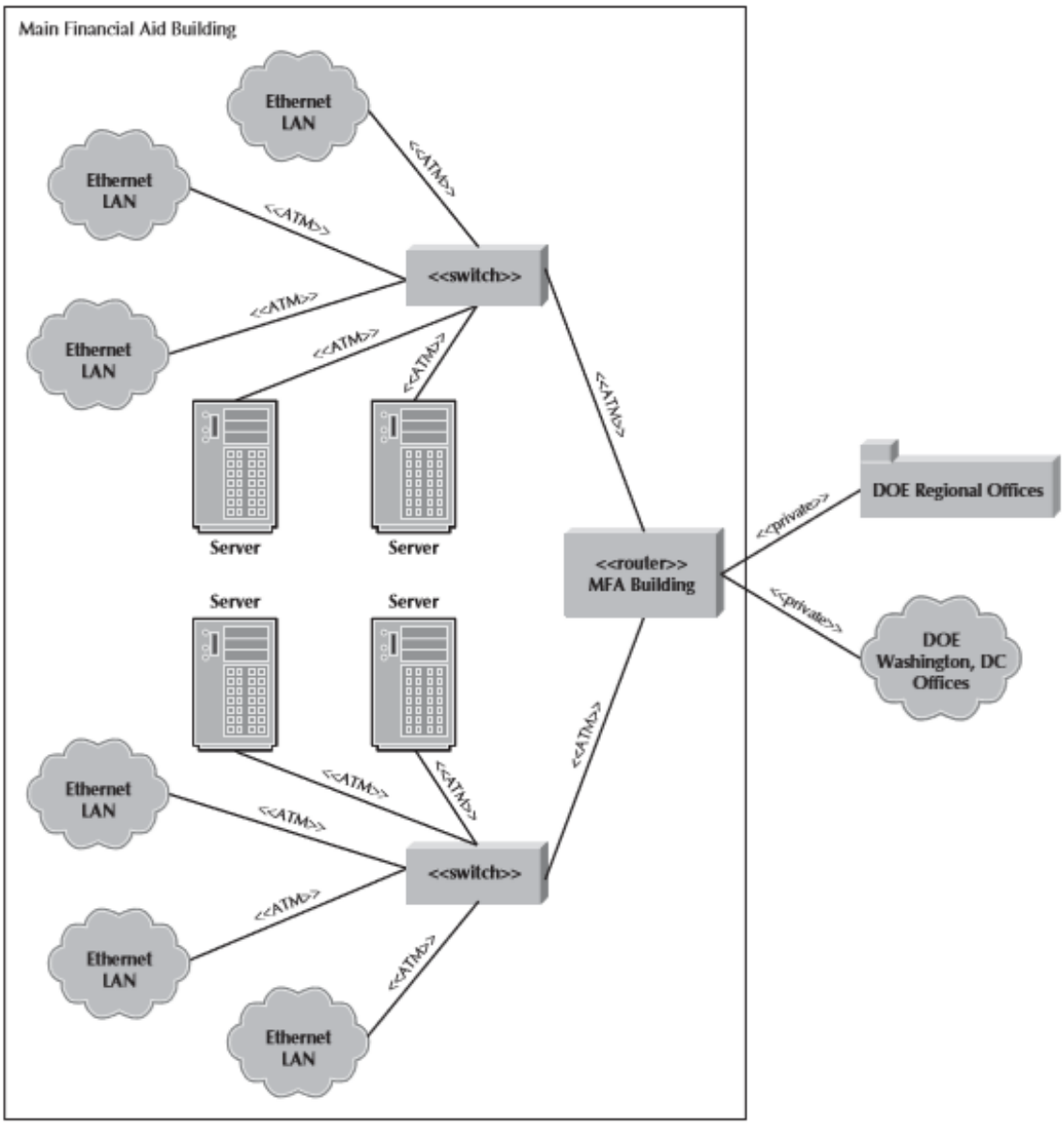
Selanjutnya digambar garis-garis yang menghubungkan komponen-komponen yang secara fisik akan melekat satu sama lain. Dalam hal perangkat lunak, beberapa model jaringan mencantumkan perangkat lunak yang diperlukan untuk setiap komponen model jaringan tepat pada diagram, sedangkan di lain waktu, perangkat lunak dijelaskan dalam memo yang dilampirkan pada model jaringan. Gambar 11-11 menunjukkan diagram penyebaran yang menggambarkan dua tingkat detail dari model jaringan tingkat rendah. Perhatikan, kami menggunakan notasi node standar dan diperpanjang pada gambar ini. Dalam hal ini, kami telah menyertakan sebuah paket (lihat Bab 7) untuk mewakili satu set koneksi ke router di gedung MFA. Dengan menyertakan paket, kami hanya menampilkan detail yang diperlukan. Notasi yang diperluas dalam banyak kasus membantu pengguna dalam memahami topologi lapisan arsitektur fisik jauh lebih baik daripada notasi standar. Sebaiknya gunakan simbol yang menyampaikan pesan dengan baik.



Gambar 11-10 Representasi Diagram Penerapan Model Jaringan Tingkat Atas



(a)



(b)

Gambar 11-11 Representasi Diagram Penerapan dari Model Jaringan Tingkat Rendah

Pengalaman kami menunjukkan bahwa sebagian besar tim proyek membuat memo untuk file proyek yang memberikan detail tambahan tentang model jaringan. Informasi ini berguna bagi orang-orang yang bertanggung jawab untuk membuat spesifikasi perangkat keras dan perangkat lunak (dijelaskan nanti dalam bab ini) dan yang akan bekerja lebih luas dengan pengembangan infrastruktur. Memo ini dapat mencakup masalah khusus yang memengaruhi komunikasi, persyaratan untuk perangkat keras dan perangkat lunak yang mungkin tidak terlihat jelas dari model jaringan, atau vendor perangkat keras atau perangkat lunak tertentu atau produk yang harus diperoleh.

Tujuan utama dari diagram model jaringan adalah untuk menyajikan infrastruktur yang diusulkan untuk sistem baru. Tim proyek dapat menggunakan diagram untuk memahami ruang lingkup sistem, kompleksitas strukturnya, masalah komunikasi penting apa pun yang mungkin memengaruhi pengembangan dan implementasi, dan komponen aktual yang perlu diperoleh atau diintegrasikan ke dalam lingkungan.

11.8 SPESIFIKASI PERANGKAT KERAS DAN PERANGKAT LUNAK SISTEM

Waktu untuk mulai memperoleh perangkat keras dan perangkat lunak yang akan dibutuhkan untuk sistem masa depan adalah selama desain sistem. Dalam banyak kasus, sistem baru hanya akan berjalan pada peralatan yang ada dalam organisasi. Namun, di lain waktu, perangkat keras dan perangkat lunak baru harus dibeli. Spesifikasi perangkat keras dan perangkat lunak adalah dokumen yang menjelaskan perangkat keras dan perangkat lunak apa yang diperlukan untuk mendukung suatu aplikasi. Akuisisi perangkat keras dan perangkat lunak yang sebenarnya harus diserahkan kepada departemen pembelian atau area dalam organisasi yang menangani pengadaan modal. Namun, tim proyek menulis spesifikasi perangkat keras dan perangkat lunak untuk mengkomunikasikan kebutuhan proyek kepada orang yang tepat. Ada beberapa langkah yang terlibat dalam pembuatan dokumen. Gambar 11-12 menunjukkan contoh spesifikasi perangkat keras dan perangkat lunak.

Pertama, kita perlu mendefinisikan perangkat lunak yang akan dijalankan pada setiap komponen. Ini biasanya dimulai dengan sistem operasi (misalnya, Windows, Linux) dan termasuk perangkat lunak tujuan khusus pada klien dan server (misalnya, database Oracle). Dokumen ini harus mempertimbangkan biaya tambahan, seperti pelatihan teknis, pemeliharaan, perpanjangan garansi, dan perjanjian lisensi (misalnya, lisensi situs untuk paket perangkat lunak). Kebutuhan yang terdaftar dipengaruhi oleh keputusan yang dibuat dalam kegiatan desain lainnya.

Specification	Standard Client	Standard Web Server	Standard Application Server	Standard Database Server
Operating System	• Windows • Internet Explorer	• Linux	• Linux	• Linux
Special Software	• Acrobat Reader • Adobe Flash • QuickTime	• Apache	• Java	• Oracle
Hardware	• 8 GB Memory • 500 GB disk drive • Intel Core i5 • 2--22″ monitors	• 16 GB Memory • 1TB disk drive • Intel Xenon E%-2400 • 1--22″ monitor	• 32 GB Memory • 2–1 TB disk drives • Intel Xenon E5-2600 • 1--22″ monitor	• 32 GB Memory • 4–1 TB Hotplug disk drives • Intel Xenon E5-2600 • 1--22″ monitor
Network	• 100 Mbps Ethernet High-speed Wireless	• 100 Mbps Ethernet	• 100 Mbps Ethernet	• 100 Mbps Ethernet

Gambar 11-12 Contoh Spesifikasi Perangkat Keras dan Perangkat Lunak

Kedua, kita harus membuat daftar perangkat keras yang diperlukan untuk mendukung sistem di masa depan. Dengan munculnya komputasi mobile (lihat Bab 10), Komputasi Cloud (lihat sebelumnya dalam bab ini), IoT (lihat sebelumnya dalam bab ini), dan Green IT (lihat sebelumnya dalam bab ini), langkah ini jauh lebih terlibat daripada itu dulu. Namun, model jaringan tingkat rendah memberikan titik awal yang baik untuk merekam kebutuhan perangkat keras proyek karena setiap komponen pada diagram sesuai dengan item dalam daftar ini. Secara umum, daftar tersebut dapat mencakup hal-hal seperti server database, server jaringan, perangkat periferal (misalnya, printer, pemindai), perangkat cadangan, komponen penyimpanan, dan komponen perangkat keras lainnya yang diperlukan untuk mendukung aplikasi. Pada saat ini, Anda juga harus mencatat jumlah setiap barang yang akan dibutuhkan.

Ketiga, kita harus menjelaskan, sedetail mungkin, persyaratan minimum untuk setiap perangkat keras. Biasanya, tim proyek harus menyampaikan persyaratan seperti jumlah kapasitas pemrosesan, jumlah ruang penyimpanan, dan fitur khusus apa pun yang harus disertakan. Banyak organisasi memiliki daftar standar perangkat keras dan perangkat lunak yang disetujui yang harus digunakan; jadi dalam banyak kasus, langkah ini hanya melibatkan pemilihan item dari daftar. Namun, di lain waktu, tim beroperasi di wilayah baru dan tidak dibatasi oleh kebutuhan untuk memilih dari daftar yang disetujui. Langkah ini menjadi lebih mudah dengan pengalaman; namun, ada beberapa petunjuk yang dapat membantu Anda menjelaskan kebutuhan perangkat keras (lihat Gambar 11-13). Misalnya, pertimbangkan standar perangkat keras dalam organisasi atau yang direkomendasikan oleh vendor. Bicaralah dengan pengembang sistem berpengalaman atau perusahaan lain dengan sistem serupa. Terakhir, pikirkan tentang faktor-faktor yang memengaruhi kinerja perangkat keras, seperti ekspektasi waktu respons pengguna, volume data, kebutuhan memori perangkat lunak, jumlah pengguna yang mengakses sistem, jumlah koneksi eksternal, dan proyeksi pertumbuhan.

Langkah terakhir yang perlu dipertimbangkan adalah mengevaluasi proposal vendor (lihat Bab 7). Cara termudah untuk melakukannya adalah dengan membuat matriks alternatif (lihat Bab 2 dan 7). Dalam hal ini, kriteria evaluasi dalam matriks alternatif harus mencakup semua persyaratan arsitektur, baik opsional maupun wajib, dan setiap kriteria harus diberi bobot. Beberapa kriteria umum termasuk kecepatan CPU, kecepatan bus, ukuran disk, waktu akses disk, ukuran cache, kecepatan cache, ukuran RAM, kecepatan RAM, kecepatan transfer data, ukuran dan kecepatan RAM video, ukuran monitor, dan resolusi printer. Tentu saja, di dunia yang terhubung saat ini, perangkat keras dan perangkat lunak jaringan juga perlu ditentukan, termasuk router, server cetak, hub, dan sakelar. Perangkat seluler seperti smartphone dan tablet dapat menjadi bagian dari solusi arsitektur fisik. Bergantung pada persyaratan domain masalah, perangkat keras tambahan dan perangkat lunak sistem mungkin diperlukan, seperti pengenalan suara dan perangkat lunak dan perangkat keras generasi, tablet digitalisasi, dan mungkin layar yang dipasang di kepala, kacamata rana, perangkat penunjuk umpan balik paksa, dan printer 3D. Masing-masing jenis perangkat khusus ini memiliki kriteria evaluasi khusus sendiri. Singkatnya, ketika membuat spesifikasi perangkat keras dan perangkat lunak sistem, sebagian besar analis sistem menemukan bahwa mereka membutuhkan bantuan dari personel TI dan CS.

Fungsi dan Fitur Fungsi dan fitur spesifik apa yang diperlukan (misalnya, ukuran monitor, fitur perangkat lunak)?

Kinerja Seberapa cepat perangkat keras dan perangkat lunak beroperasi (misalnya, prosesor, jumlah penulisan database per detik)?

Database dan Sistem Warisan Seberapa baik perangkat keras dan perangkat lunak berinteraksi dengan sistem lama (misalnya, dapatkah ia menulis ke database ini)?

Strategi Perangkat Keras dan OS Apa rencana migrasi ke depan (misalnya, tujuannya adalah untuk memiliki semua peralatan satu vendor)?

Biaya Kepemilikan Berapa biaya di luar pembelian (misalnya, biaya lisensi tambahan, pemeliharaan tahunan, biaya pelatihan, biaya gaji)?

Preferensi Politik Orang adalah makhluk kebiasaan dan resisten terhadap perubahan, sehingga perubahan harus diminimalkan.

Kinerja Vendor Beberapa vendor memiliki reputasi atau prospek masa depan yang berbeda dari sistem perangkat keras atau perangkat lunak tertentu yang mereka jual saat ini.

Gambar 11-13 Faktor Pemilihan Hardware dan Software

Tergantung pada keseluruhan biaya dan ukuran proyek, satu hal yang harus dipertimbangkan secara serius adalah penggunaan benchmark. Benchmark pada dasarnya adalah contoh program yang diharapkan berjalan pada arsitektur fisik baru. Meskipun benchmark bisa mahal untuk dibuat, mereka cenderung memberikan gambaran yang lebih realistis tentang bagaimana lapisan arsitektur fisik yang diusulkan akan tampil.

Saat mengevaluasi perangkat keras, ada serangkaian masalah yang harus Anda kenali.

- Anda tidak hanya harus memberikan contoh program untuk tolok ukur, tetapi Anda juga perlu memberikan data aktual. Jika tidak, hasil benchmark bisa menyesatkan.
- Anda perlu hati-hati meninjau campuran perangkat lunak dan perangkat keras sistem. Misalnya, dalam banyak kasus, Linux berkinerja lebih baik pada perangkat keras yang sama jika dibandingkan dengan Windows, tetapi beberapa aplikasi mungkin tidak tersedia di Linux. Akibatnya, mungkin ada beberapa trade-off yang harus dipertimbangkan.
- Saat mempertimbangkan untuk menambahkan perangkat keras tambahan, pastikan untuk mengevaluasi perangkat keras tambahan berdasarkan utilitas marginal, bukan utilitas sebenarnya.
- Jangan tentukan arsitektur fisik sebelum Anda memahami persyaratan domain masalah. Ini mungkin tampak jelas, tetapi ketika Anda mempertimbangkan waktu yang diperlukan untuk komputer mainframe, sejumlah besar server, atau sejumlah besar mesin klien untuk ditentukan, dipesan, dan dikirim, mungkin tergoda untuk menentukan perangkat keras dan sistem. perangkat lunak sebelum waktunya. Ini dapat menyebabkan spesifikasi di bawah atau di atas.
- Rmengenali realitas Hukum Parkinson. Dari perspektif TI, Hukum Parkinson menyiratkan bahwa terlepas dari kebutuhan nyata pengguna, kebutuhan imajiner mereka akan selalu memenuhi kapasitas apa pun yang dimiliki sistem. Akibatnya, sangat penting bahwa desain lapisan arsitektur fisik didasarkan pada arsitektur masa depan saat ini dan yang diharapkan dari lapisan domain masalah.
- Jangan membatasi pilihan pada satu vendor. Ini terutama benar ketika Anda mempertimbangkan perangkat keras komoditas, seperti layar, desktop, dan server ukuran departemen.
- Mengingat tingkat perubahan teknologi yang terjadi di TI, pertimbangkan ide-ide terdepan. Misalnya, meskipun komputer tablet sudah ada sejak lama, iPad™ tidak ada dalam radar kebanyakan orang. Saat ini, ini dianggap sebagai pengubah permainan ketika mempertimbangkan perangkat keras berbasis klien. Akibatnya, Anda benar-benar harus tetap up to date dalam hal desain lapisan arsitektur fisik.

11.9 PERSYARATAN NONFUNGSIONAL DAN DESAIN LAPISAN ARSITEKTUR FISIK

Perancangan lapisan arsitektur fisik menentukan keseluruhan arsitektur dan penempatan perangkat lunak dan perangkat keras yang akan digunakan. Masing-masing arsitektur yang dibahas sebelumnya memiliki kekuatan dan kelemahan. Sebagian besar organisasi menggunakan arsitektur client-server karena alasan biaya, jadi jika tidak ada alasan kuat untuk memilih satu arsitektur di atas yang lain, biaya biasanya menyarankan client-server.

Membuat desain lapisan arsitektur fisik dimulai dengan persyaratan nonfungsional. Langkah pertama adalah menyempurnakan kebutuhan nonfungsional menjadi kebutuhan yang lebih detail yang kemudian digunakan untuk membantu memilih arsitektur yang akan digunakan (berbasis server, berbasis klien, atau klien-server) dan komponen perangkat lunak apa yang akan ditempatkan pada setiap perangkat. . Dalam arsitektur client-server, kita juga harus memutuskan apakah akan menggunakan arsitektur two-tier, three-tier, atau n-tier. Kemudian kebutuhan nonfungsional dan desain arsitektur digunakan untuk mengembangkan spesifikasi perangkat keras dan perangkat lunak.

Empat jenis utama kebutuhan nonfungsional dapat menjadi penting dalam merancang arsitektur: persyaratan operasional, persyaratan kinerja, persyaratan keamanan, dan persyaratan budaya/politik. Selanjutnya, masing-masing persyaratan ini harus sepenuhnya diverifikasi dan divalidasi.

Kebutuhan operasional

Persyaratan operasional menentukan lingkungan operasi di mana sistem harus melakukan dan bagaimana mereka mungkin berubah dari waktu ke waktu. Ini biasanya mengacu pada sistem operasi, perangkat lunak sistem, dan sistem informasi yang harus berinteraksi dengan sistem, tetapi kadang-kadang juga mencakup lingkungan fisik jika lingkungan penting bagi aplikasi (misalnya, terletak di lantai pabrik yang bising, jadi tidak peringatan yang dapat didengar dapat didengar). Gambar 11-14 merangkum empat bidang persyaratan operasional utama dan memberikan beberapa contoh masing-masing.

Persyaratan Lingkungan Teknis. Persyaratan lingkungan teknis menentukan jenis sistem perangkat keras dan perangkat lunak tempat sistem akan bekerja. Persyaratan ini biasanya berfokus pada perangkat lunak sistem operasi (misalnya, Windows, Linux, Mac OS), perangkat lunak sistem database (misalnya, Oracle), dan perangkat lunak sistem lainnya (misalnya, Firefox). Di dunia terdistribusi saat ini, masalah yang terkait dengan komputasi seluler (lihat Bab 10), Komputasi Cloud (lihat bagian sebelumnya dalam bab ini), IoT (lihat bagian sebelumnya dalam bab ini), dan TI Hijau (lihat bagian sebelumnya dalam bab ini).) sangat relevan. Akibatnya, ini juga mencakup semua jenis perangkat keras yang berbeda dari komputer mainframe hingga smartphone. Bergantung pada aplikasi yang diterapkan pada arsitektur fisik, perangkat keras khusus mungkin diperlukan, seperti tampilan 3D, pencetakan 3D, sistem suara 3D, dan tablet dengan akselerometer. Dengan teknologi saat ini, kemungkinan kombinasi perangkat keras yang dapat digunakan untuk memecahkan masalah hampir tidak ada habisnya. Akibatnya, ini adalah salah satu bidang di mana keahlian tambahan mungkin diperlukan.

Persyaratan Integrasi Sistem. Persyaratan integrasi sistem adalah persyaratan yang mengharuskan sistem untuk beroperasi dengan sistem informasi lain, baik di dalam maupun di luar perusahaan. Ini biasanya menentukan antarmuka melalui mana data akan dipertukarkan dengan sistem lain.

Jenis Permintaan	Definisi	Contoh
Persyaratan Lingkungan Teknis	Persyaratan perangkat keras, perangkat lunak, dan jaringan khusus yang dikenakan oleh persyaratan bisnis	• Sistem akan bekerja melalui lingkungan Web dengan Internet Explorer. • Semua lokasi kantor akan memiliki koneksi jaringan yang selalu aktif untuk mengaktifkan pembaruan basis data waktu nyata. • Versi sistem akan disediakan untuk pelanggan yang terhubung melalui Internet melalui tablet atau smartphone.
Persyaratan Integrasi Sistem	Sejauh mana sistem akan beroperasi dengan sistem lain	• Sistem harus dapat mengimpor dan mengekspor spreadsheet Excel. • Sistem akan membaca dan menulis ke database persediaan utama dalam sistem persediaan.
Persyaratan Portabilitas	Sejauh mana sistem perlu beroperasi di lingkungan lain	• Sistem harus dapat bekerja dengan sistem operasi yang berbeda (misalnya, Linux, Mac OS, dan Windows). • Sistem mungkin perlu beroperasi dengan perangkat genggam, seperti perangkat Android dan Apple iOS.
Persyaratan Pemeliharaan	Perubahan bisnis yang diharapkan dimana sistem harus dapat beradaptasi	• Sistem ini akan dapat mendukung lebih dari satu pabrik dengan pemberitahuan enam bulan sebelumnya. • Versi baru dari sistem akan dirilis setiap enam bulan

Gambar 11-14 Persyaratan Operasional

Persyaratan Portabilitas. Sistem informasi tidak pernah tetap konstan. Bisnis membutuhkan perubahan dan teknologi operasi berubah, sehingga sistem informasi yang mendukung dan menjalankannya juga harus berubah. Persyaratan portabilitas menentukan bagaimana lingkungan operasi teknis dapat berubah dari waktu ke waktu dan bagaimana sistem harus merespons (misalnya, sistem saat ini berjalan di Windows, sedangkan di masa depan sistem mungkin harus digunakan di Linux). Persyaratan portabilitas juga mengacu pada perubahan potensial dalam persyaratan bisnis yang mendorong perubahan lingkungan teknis. Misalnya, di masa mendatang pengguna mungkin ingin mengakses situs web dari ponsel mereka.

Persyaratan Pemeliharaan. Persyaratan rawatan menentukan perubahan persyaratan bisnis yang dapat diantisipasi. Tidak semua perubahan dapat diprediksi, tetapi beberapa dapat diprediksi. Misalnya, sebuah perusahaan kecil hanya memiliki satu pabrik tetapi mengantisipasi pembangunan pabrik kedua dalam lima tahun ke depan. Semua sistem informasi harus ditulis untuk memudahkan pelacakan setiap pabrik secara terpisah, baik untuk

sistem personalia, penganggaran, atau inventaris. Persyaratan rawatan berusaha untuk mengantisipasi persyaratan masa depan sehingga sistem yang dirancang hari ini akan mudah dipelihara jika dan ketika persyaratan masa depan itu muncul. Persyaratan rawatan juga dapat menentukan siklus pembaruan untuk sistem, seperti frekuensi rilis versi baru.

Persyaratan Kinerja

Persyaratan kinerja fokus pada masalah kinerja, seperti waktu respons, kapasitas, dan keandalan. Gambar 11-15 merangkum tiga bidang persyaratan kinerja utama dan memberikan beberapa contoh.

Persyaratan Kecepatan. Persyaratan kecepatan persis seperti yang mereka katakan: Seberapa cepat sistem harus beroperasi? Pertama adalah waktu respon sistem: Berapa lama waktu yang dibutuhkan sistem untuk menanggapi permintaan pengguna. Meskipun setiap orang lebih menyukai waktu respons yang rendah, dengan sistem yang segera merespons setiap permintaan pengguna, hal ini tidak praktis. Kita bisa merancang sistem seperti itu, tapi itu akan mahal. Sebagian besar pengguna memahami bahwa bagian tertentu dari suatu sistem akan merespons dengan cepat, sedangkan yang lain lebih lambat. Tindakan yang dilakukan secara lokal di komputer pengguna harus segera (misalnya, mengetik, menyeret, dan menjatuhkan), sedangkan tindakan lain yang memerlukan komunikasi melalui jaringan dapat memiliki waktu respons yang lebih lama (misalnya, permintaan Web).

Aspek kedua dari persyaratan kecepatan adalah berapa lama waktu yang dibutuhkan transaksi di satu bagian sistem untuk tercermin di bagian lain. Misalnya, seberapa cepat setelah pesanan dilakukan, item yang ada di dalamnya akan ditampilkan sebagai tidak lagi tersedia untuk dijual kepada orang lain? Jika persediaan tidak segera diperbarui, maka orang lain dapat memesan barang yang sama, hanya untuk mengetahui kemudian kehabisan stok. Hal ini terutama benar ketika kita mempertimbangkan database NoSQL yang tidak segera memperbarui semua salinan data (lihat Bab 9). Atau seberapa cepat setelah pesanan ditempatkan, pesanan dikirim ke gudang untuk diambil dari inventaris dan dikirim?

Persyaratan Kapasitas. Persyaratan kapasitas mencoba memprediksi berapa banyak pengguna yang harus didukung sistem, baik secara total maupun secara bersamaan. Persyaratan kapasitas penting dalam memahami ukuran database, kekuatan pemrosesan yang dibutuhkan, dan segera. Persyaratan paling penting biasanya adalah jumlah puncak pengguna simultan karena hal ini berdampak langsung pada kekuatan pemrosesan komputer yang diperlukan untuk mendukung sistem.

Seringkali lebih mudah untuk memprediksi jumlah pengguna untuk sistem internal yang dirancang untuk mendukung karyawan organisasi itu sendiri daripada memprediksi jumlah pengguna untuk sistem yang berhadapan dengan pelanggan, terutama yang ada di Web. Bagaimana Weather.com memperkirakan jumlah puncak pengguna yang akan mencari informasi cuaca secara bersamaan? Ini adalah seni dan sains, sehingga sering kali tim memberikan rentang perkiraan, dengan rentang yang lebih luas digunakan untuk menandakan perkiraan yang kurang akurat.

Jenis Persyaratan	Definisi	Contoh
Persyaratan Kecepatan	Waktu di mana sistem harus menjalankan fungsinya	• Waktu respons harus kurang dari 7 detik untuk setiap transaksi melalui jaringan. • Database inventaris harus diperbarui secara real time. • Pesanan akan dikirimkan ke lantai pabrik setiap 30 menit.

Persyaratan Kapasitas	Jumlah total dan puncak pengguna dan volume data yang diharapkan	• Akan ada maksimum 100–200 pengguna simultan pada waktu penggunaan puncak. • Transaksi tipikal akan membutuhkan transmisi 10 ribu data.
Persyaratan Ketersediaan dan Reliabilitas	Sejauh mana sistem akan tersedia untuk pengguna dan tingkat kegagalan yang diperbolehkan karena kesalahan	• Sistem akan menyimpan data untuk sekitar 5.000 pelanggan dengan total sekitar 2 MB data. • Pemeliharaan terjadwal tidak boleh melebihi satu periode 6 jam setiap bulan. • Sistem harus memiliki kinerja uptime 99%.

Gambar 11-15 Persyaratan Kinerja

Persyaratan Ketersediaan dan Keandalan. Persyaratan ketersediaan dan keandalan berfokus pada sejauh mana pengguna dapat mengasumsikan bahwa sistem akan tersedia untuk mereka gunakan. Meskipun beberapa sistem dimaksudkan untuk digunakan hanya selama empat puluh jam kerja seminggu, beberapa sistem dirancang untuk digunakan oleh orang-orang di seluruh dunia. Untuk sistem seperti itu, anggota tim proyek perlu mempertimbangkan bagaimana aplikasi dapat dioperasikan, didukung, dan dipelihara 24/7 (yaitu, 24 jam sehari, 7 hari seminggu). Persyaratan 24/7 ini berarti bahwa pengguna mungkin memerlukan bantuan atau memiliki pertanyaan kapan saja, dan meja dukungan yang tersedia delapan jam sehari tidak akan cukup mendukung. Penting juga untuk mempertimbangkan keandalan apa yang dibutuhkan dalam sistem. Sebuah sistem yang membutuhkan keandalan tinggi (misalnya, perangkat medis atau saklar telepon) membutuhkan perencanaan dan pengujian yang jauh lebih besar daripada yang tidak memiliki kebutuhan keandalan yang tinggi (misalnya, sistem personel, katalog Web).

Lebih sulit untuk memprediksi puncak dan lembah dalam penggunaan sistem ketika sistem memiliki khalayak global. Biasanya, aplikasi dicadangkan pada akhir pekan atau larut malam saat pengguna tidak lagi mengakses sistem. Kegiatan pemeliharaan seperti itu perlu dipikirkan kembali dengan inisiatif global. Misalnya, hari apa dalam seminggu yang dianggap sebagai hari "turun". Di berbagai belahan dunia, bisnis tidak berlangsung setiap hari. Di beberapa bagian, Jumat itu suci; di bagian lain, itu hari Sabtu atau Minggu. Akibatnya, masalah politik dan budaya (dijelaskan di bawah dan di Bab 10) dapat memengaruhi persyaratan kinerja. Pengembangan antarmuka Web, khususnya, telah meningkatkan kebutuhan akan dukungan 24/7; secara default, Web dapat diakses oleh siapa saja kapan saja. Misalnya, para pengembang aplikasi Web untuk peritel pakaian dan perlengkapan luar ruang AS, Orvis, terkejut ketika pesanan pertama setelah ditayangkan datang dari Jepang.

Persyaratan Keamanan

Keamanan adalah kemampuan untuk melindungi sistem informasi dari gangguan dan kehilangan data, baik yang disebabkan oleh tindakan yang disengaja (misalnya, peretas, serangan teroris) atau peristiwa acak (misalnya, kegagalan disk, tornado). Keamanan terutama merupakan tanggung jawab kelompok operasi—staf yang bertanggung jawab untuk memasang dan mengoperasikan kontrol keamanan, seperti firewall, sistem deteksi intrusi, dan operasi pencadangan dan pemulihan rutin. Meskipun demikian, pengembang sistem baru harus memastikan bahwa persyaratan keamanan sistem menghasilkan tindakan pencegahan yang wajar untuk mencegah masalah; pengembang sistem bertanggung jawab untuk memastikan keamanan dalam sistem informasi itu sendiri.

Keamanan adalah masalah yang terus meningkat di dunia yang mendukung Internet saat ini. Secara historis, ancaman keamanan terbesar datang dari dalam organisasi itu sendiri. Sejak awal 1980-an ketika FBI pertama kali mulai menyimpan statistik kejahatan komputer dan perusahaan keamanan mulai melakukan survei kejahatan komputer, karyawan organisasi telah melakukan sebagian besar kejahatan komputer. Selama bertahun-tahun, 80 persen pembobolan, pencurian, dan sabotase yang tidak sah telah dilakukan oleh orang dalam, hanya menyisakan 20 persen untuk peretas di luar organisasi.

Pada tahun 2001, itu berubah. Bergantung pada survei apa yang Anda baca, persentase insiden yang dikaitkan dengan peretas eksternal pada tahun 2001 meningkat menjadi 50 hingga 70 persen dari semua insiden, yang berarti bahwa risiko terbesar yang dihadapi organisasi sekarang adalah dari luar. Meskipun beberapa dari perubahan ini mungkin disebabkan oleh keamanan internal yang lebih baik dan komunikasi yang lebih baik dengan karyawan untuk mencegah masalah keamanan, sebagian besar hanya karena peningkatan aktivitas oleh peretas eksternal. Dengan Komputasi Cloud dan IoT, keamanan menjadi lebih penting.

Mengembangkan persyaratan keamanan biasanya dimulai dengan beberapa penilaian nilai sistem dan datanya. Ini membantu menentukan sistem yang sangat penting sehingga staf operasi sadar akan risikonya. Keamanan dalam sistem biasanya berfokus pada menentukan siapa yang dapat mengakses data apa, mengidentifikasi kebutuhan enkripsi dan otentikasi, dan memastikan aplikasi mencegah penyebaran virus (lihat Gambar 11-16).

Nilai Sistem. Aset komputer yang paling penting dalam organisasi mana pun bukanlah peralatannya; itu adalah data organisasi. Misalnya, seseorang menghancurkan komputer mainframe senilai Rp. 15 miliar. Mainframe bisa diganti, cukup dengan membeli yang baru. Itu akan mahal, tetapi masalahnya akan terpecahkan dalam beberapa minggu. Sekarang anggaplah seseorang menghancurkan semua catatan mahasiswa di universitas Anda sehingga tidak ada yang tahu mata kuliah apa yang diambil atau nilai mereka. Biayanya akan jauh melebihi biaya penggantian komputer senilai Rp. 15 miliar. Tuntutan hukum saja akan dengan mudah melebihi Rp. 15 miliar, dan biaya staf untuk menemukan catatan kertas dan memasukkan kembali data dari mereka akan sangat besar dan tentu saja akan memakan waktu lebih dari beberapa minggu.

Dalam beberapa kasus, sistem informasi itu sendiri memiliki nilai yang jauh melebihi biaya peralatan juga. Misalnya, untuk bank Internet yang tidak memiliki cabang bata dan mortir, situs web adalah sistem yang sangat penting. Jika situs web rusak, bank tidak dapat melakukan bisnis dengan pelanggannya. Sebuah aplikasi mission-critical adalah sistem informasi yang secara harfiah penting untuk kelangsungan hidup organisasi. Ini adalah aplikasi yang tidak boleh gagal, dan jika gagal, staf jaringan akan menghapus semua yang lain untuk memperbaikinya. Aplikasi mission-critical biasanya diidentifikasi dengan jelas sehingga kepentingannya tidak diabaikan.

Jenis Persyaratan	Definisi	Contoh
Estimasi Nilai Sistem	Perkiraan nilai bisnis sistem dan datanya	• Sistem ini tidak memiliki misi yang kritis, tetapi pemadaman sistem diperkirakan menelan biaya 650 juta rupiah per jam dari pendapatan yang hilang. • Hilangnya seluruh data sistem diperkirakan menelan biaya 260 miliar rupiah.

Persyaratan Kontrol Akses	Batasan siapa yang dapat mengakses data apa	• Hanya manajer departemen yang dapat mengubah item inventaris di dalam departemen mereka sendiri. • Operator telepon dapat membaca dan membuat item dalam file pelanggan tetapi tidak dapat mengubah atau menghapus item.
Enkripsi dan Persyaratan Otentikasi	Mendefinisikan data apa yang akan dienkripsi di mana dan apakah otentikasi akan diperlukan untuk akses pengguna	• Data akan dienkripsi dari komputer pengguna ke situs web untuk memberikan pesanan yang aman • Pengguna yang masuk dari luar kantor akan diperlukan untuk otentikasi.
Persyaratan Pengendalian Virus	Persyaratan untuk mengendalikan penyebaran virus	• Semua file yang diunggah akan diperiksa virusnya sebelum disimpan di sistem.

Gambar 11-16 Persyaratan Keamanan

Bahkan gangguan sementara dalam layanan dapat menimbulkan biaya yang signifikan. Biaya gangguan pada situs web utama perusahaan atau LAN dan tulang punggung yang mendukung operasi penjualan telepon sering diukur dalam jutaan dolar. Amazon.com, misalnya, memiliki pendapatan lebih dari Rp. 15 miliar juta per jam, jadi jika situs webnya tidak tersedia selama satu jam atau bahkan sebagian dari satu jam, ia akan kehilangan pendapatan jutaan dolar. Perusahaan yang melakukan lebih sedikit e-bisnis atau melakukan penjualan telepon memiliki biaya yang lebih rendah, tetapi survei terbaru menunjukkan kerugian sebesar Rp 15.000.000.000 hingga Rp. 3.000.000.000 per jam tidak jarang terjadi pada sistem informasi utama yang dihadapi pelanggan.

Persyaratan Kontrol Akses. Beberapa data yang disimpan dalam sistem perlu dirahasiakan; beberapa data memerlukan kontrol khusus tentang siapa yang diizinkan untuk mengubah atau menghapusnya. Catatan personalia, misalnya, harus dapat dibaca hanya oleh departemen personalia dan supervisor karyawan; perubahan harus diizinkan untuk dilakukan hanya oleh departemen personalia. Persyaratan kontrol akses menyatakan siapa yang dapat mengakses data apa dan jenis akses apa yang diizinkan: apakah individu dapat membuat, membaca, memperbarui, dan/atau menghapus data. Persyaratan mengurangi kemungkinan bahwa pengguna sistem yang berwenang dapat melakukan tindakan yang tidak sah. Salah satu pendekatan untuk mengatasi persyaratan ini adalah melalui penggunaan daftar kontrol akses, yang dapat diimplementasikan melalui sistem operasi atau sistem manajemen database.

Persyaratan Enkripsi dan Otentikasi. Salah satu cara terbaik untuk mencegah akses tidak sah ke data adalah enkripsi, yang merupakan sarana untuk menyamarkan informasi dengan menggunakan algoritma matematika (atau rumus). Enkripsi dapat digunakan untuk melindungi data yang disimpan dalam database atau data yang sedang transit melalui jaringan dari database ke komputer. Ada dua jenis enkripsi yang berbeda secara mendasar: simetris dan asimetris. Algoritma enkripsi simetris seperti *Data Encryption Standard* (DES) atau *Advanced Encryption Standard* (AES)] adalah algoritma di mana utama yang digunakan untuk mengenkripsi pesan sama dengan utama yang digunakan untuk mendekripsi, yang berarti penting untuk melindungi utama dan bahwa utama terpisah harus digunakan untuk setiap orang atau organisasi yang berbagi informasi dengan sistem (atau semua orang dapat membaca semua data).

Sebuah algoritma enkripsi asimetris (seperti enkripsi utama publik) adalah salah satu di mana utama yang digunakan untuk mengenkripsi data (disebut utama publik) berbeda dari yang digunakan untuk mendekripsi (disebut utama pribadi). Bahkan jika semua orang mengetahui utama publik, setelah data dienkripsi, mereka tidak dapat didekripsi tanpa utama pribadi. Enkripsi utama publik sangat mengurangi masalah manajemen utama. Setiap pengguna memiliki utama publik yang digunakan untuk mengenkripsi pesan yang dikirim ke sana. Utama publik ini dipublikasikan secara luas (misalnya, tercantum dalam direktori gaya buku telepon)—itulah sebabnya mereka disebut utama publik. Utama pribadi, sebaliknya, dirahasiakan (itulah sebabnya disebut pribadi).

Enkripsi utama publik juga mengizinkan otentikasi (atau tanda tangan digital). Ketika satu pengguna mengirim pesan ke yang lain, sulit untuk membuktikan secara hukum siapa yang sebenarnya mengirim pesan. Bukti hukum penting dalam banyak komunikasi, seperti transfer bank dan pesanan beli/jual dalam mata uang dan perdagangan saham, yang biasanya memerlukan tanda tangan yang sah. Algoritme enkripsi utama publik dapat dibalik, artinya teks yang dienkripsi dengan salah satu utama dapat didekripsi oleh utama lainnya. Biasanya, kami mengenkripsi dengan utama publik dan mendekripsi dengan utama pribadi. Namun, dimungkinkan untuk melakukan sebaliknya: mengenkripsi dengan utama pribadi dan mendekripsi dengan utama publik. Karena utama pribadi bersifat rahasia, hanya pengguna sebenarnya yang dapat menggunakannya untuk mengenkripsi pesan. Dengan demikian, tanda tangan digital atau urutan otentikasi digunakan sebagai tanda tangan legal pada banyak transaksi keuangan. Tanda tangan ini biasanya nama pihak yang menandatangani ditambah informasi unik lainnya dari pesan (misalnya, tanggal, waktu, atau jumlah dolar). Tanda tangan ini dan informasi lainnya dienkripsi oleh pengirim menggunakan utama pribadi. Penerima menggunakan utama publik pengirim untuk mendekripsi blok tanda tangan dan membandingkan hasilnya dengan nama dan isi utama lainnya di sisa pesan untuk memastikan kecocokan.

Satu-satunya masalah dengan pendekatan ini terletak pada memastikan bahwa orang atau organisasi yang mengirim dokumen dengan utama pribadi yang benar adalah orang atau organisasi yang sebenarnya. Siapa pun dapat memposting utama publik di Internet, jadi tidak ada cara untuk mengetahui dengan pasti siapa yang sebenarnya menggunakannya. Misalnya, mungkin saja seseorang selain Organisasi A dalam contoh ini mengklaim sebagai Organisasi A padahal sebenarnya dia adalah seorang penipu.

Di sinilah infrastruktur utama publik Internet/ *public key infrastructure* (PKI) menjadi penting. PKI adalah seperangkat perangkat keras, perangkat lunak, organisasi, dan kebijakan yang dirancang untuk membuat enkripsi utama publik berfungsi di Internet. PKI dimulai dengan otoritas sertifikat*/ certifi cate authorit* (CA), yang merupakan organisasi tepercaya yang dapat menjamin keaslian orang atau organisasi menggunakan otentikasi (mis., VeriSign). Seseorang yang ingin menggunakan CA mendaftar dengan CA dan harus memberikan beberapa bukti identitas. Ada beberapa tingkat sertifikasi, mulai dari konfirmasi sederhana dari alamat email yang valid hingga pemeriksaan latar belakang ala polisi lengkap dengan wawancara langsung. CA mengeluarkan sertifikat digital yang merupakan utama publik pemohon, dienkripsi menggunakan utama pribadi CA sebagai bukti identifikasi. Sertifikat ini kemudian dilampirkan ke email pengguna atau transaksi Web selain informasi otentikasi. Penerima kemudian memverifikasi sertifikat dengan mendekripsi dengan utama publik CA dan juga harus menghubungi CA untuk memastikan bahwa sertifikat pengguna belum dicabut oleh CA.

Persyaratan enkripsi dan otentikasi menyatakan persyaratan enkripsi dan otentikasi apa yang diperlukan untuk data apa. Misalnya, apakah data sensitif seperti nomor kartu kredit pelanggan akan disimpan dalam database dalam bentuk terenkripsi, atau akankah enkripsi

digunakan untuk menerima pesanan melalui Internet dari situs web perusahaan? Apakah pengguna akan diminta untuk menggunakan sertifikat digital selain kata sandi standar?

Persyaratan Pengendalian Virus. Persyaratan pengendalian virus mengatasi satu-satunya masalah keamanan yang paling umum: virus. Penelitian telah menunjukkan bahwa hampir 90 persen organisasi menderita infeksi virus setiap tahun. Virus menyebabkan kejadian yang tidak diinginkan—beberapa tidak berbahaya (seperti pesan pengganggu), beberapa serius (seperti penghancuran data). Setiap kali sistem mengizinkan data untuk diimpor atau diunggah dari komputer pengguna, ada potensi infeksi virus. Banyak sistem mengharuskan semua sistem informasi yang mengizinkan impor atau pengunggahan file pengguna untuk memeriksa file tersebut dari virus sebelum disimpan dalam sistem.

Persyaratan Budaya dan Politik

Persyaratan budaya dan politik adalah persyaratan khusus untuk negara di mana sistem akan digunakan. Dalam lingkungan bisnis global saat ini, organisasi memperluas sistem mereka untuk menjangkau pengguna di seluruh dunia. Meskipun hal ini dapat masuk akal secara bisnis, dampaknya terhadap pengembangan aplikasi tidak boleh diremehkan. Namun bagian penting lain dari desain arsitektur fisik sistem adalah memahami persyaratan budaya dan politik global untuk sistem (lihat Bab 10 dan Gambar 11-17).

Persyaratan Kustomisasi. Untuk aplikasi global, tim proyek perlu memikirkan beberapa persyaratan penyesuaian: Berapa banyak aplikasi yang akan dikendalikan oleh grup pusat, dan berapa banyak aplikasi yang akan dikelola secara lokal? Misalnya, beberapa perusahaan mengizinkan anak perusahaan di beberapa negara untuk menyesuaikan aplikasi dengan menghilangkan atau menambahkan fitur tertentu. Keputusan ini memiliki trade-off antara fleksibilitas dan kontrol karena kustomisasi sering mempersulit tim proyek untuk membuat dan memelihara aplikasi. Ini juga berarti bahwa pelatihan dapat berbeda di antara berbagai bagian organisasi, dan penyesuaian dapat menimbulkan masalah ketika staf berpindah dari satu lokasi ke lokasi lain.

Karena penggunaan bahasa yang berbeda, dalam beberapa kasus, perangkat keras khusus yang telah disesuaikan dengan budaya lokal diperlukan. Misalnya, memiliki papan ketik khusus masuk akal untuk bahasa apa pun yang tidak menggunakan alfabet Romawi biasa, misalnya, Indonesia, Arab, Ibrani, Yunani, Jepang, Korea, Mandarin, atau Rusia. Ada juga emulator yang tersedia untuk berbagai bahasa. Tergantung pada pengguna yang dilayani, perangkat bantu mungkin diperlukan, seperti perangkat Braille, perangkat pelacak mata, penunjuk kepala, keyboard tongkat kepala/mulut, atau sakelar kemampuan adaptif. Tergantung pada persyaratan budaya dan politik, banyak platform perangkat keras yang berbeda mungkin perlu dipertimbangkan.

Jenis Persyaratan	Definisi	Contoh
Persyaratan Kustomisasi	Spesifikasi dari suatu aspek sistem yang dapat diubah oleh pengguna lokal	• Manajer negara akan dapat menentukan bidang baru dalam database produk untuk menangkap informasi spesifik negara • Manajer negara akan dapat mengubah format bidang nomor telepon di database pelanggan.
Persyaratan Resmi	Hukum dan peraturan yang memaksakan persyaratan pada sistem	• Informasi pribadi tentang pelanggan tidak dapat ditransfer dari negara-negara Uni Eropa ke Amerika Serikat.

		• Hal ini melanggar hukum federal AS untuk membocorkan informasi tentang siapa yang menyewa rekaman video apa, jadi akses ke riwayat sewa pelanggan hanya diizinkan untuk manajer regional.

Gambar 11-17 Persyaratan Budaya dan Politik

Persyaratan resmi. Persyaratan hukum adalah persyaratan yang diberlakukan oleh undang-undang dan peraturan pemerintah. Pengembang sistem terkadang lupa memikirkan peraturan hukum; sayangnya, melupakan memiliki beberapa risiko karena ketidaktahuan akan hukum bukanlah pembelaan. Informasi di server Web kampus terutama dalam bahasa Inggris karena kelas dilakukan dalam bahasa Inggris, yang melanggar hukum yang mengharuskan bahasa Prancis menjadi bahasa utama di semua server Internet di Prancis. Dengan secara formal mempertimbangkan peraturan hukum, Anda cenderung mengabaikannya. Contoh besar lainnya adalah keputusan pengadilan Eropa baru-baru ini mengenai hak pengguna untuk dilupakan.

Sinopsis

Dalam banyak kasus, persyaratan lingkungan teknis yang didorong oleh persyaratan bisnis dapat dengan mudah mendefinisikan lapisan arsitektur fisik. Dalam hal ini, pilihannya sederhana: Persyaratan bisnis mendominasi pertimbangan lain. Misalnya, persyaratan bisnis mungkin menentukan bahwa sistem perlu bekerja melalui Web menggunakan browser Web pelanggan. Dalam hal ini, arsitektur mungkin harus menjadi thin client-server. Persyaratan bisnis seperti itu kemungkinan besar dalam sistem yang dirancang untuk mendukung pelanggan eksternal. Sistem internal juga dapat memaksakan persyaratan bisnis, tetapi biasanya tidak terlalu membatasi.

Jika persyaratan lingkungan teknis tidak menetapkan arsitektur tertentu, maka persyaratan nonfungsional lainnya menjadi penting. Bahkan dalam kasus ketika persyaratan bisnis mendorong arsitektur, masih penting untuk bekerja melalui dan menyempurnakan persyaratan nonfungsional yang tersisa karena mereka penting dalam tahap desain dan implementasi selanjutnya.

Kebutuhan operasional. Persyaratan integrasi sistem dapat menyebabkan satu arsitektur dipilih di atas yang lain, tergantung pada arsitektur dan desain sistem yang perlu diintegrasikan dengan sistem. Misalnya, jika sistem harus terintegrasi dengan sistem desktop (misalnya, Excel), ini mungkin menyarankan arsitektur client-server yang tipis atau tebal, sedangkan jika harus terintegrasi dengan sistem berbasis server, arsitektur berbasis server dapat diindikasikan. . Sistem yang memiliki persyaratan portabilitas yang luas cenderung paling cocok untuk arsitektur thin client-server karena lebih mudah untuk menulis untuk standar berbasis Web (misalnya, HTML, XML) yang memperluas jangkauan sistem ke platform lain, daripada mencoba untuk menulis dan menulis ulang logika presentasi yang luas untuk berbagai platform dalam arsitektur berbasis server, berbasis klien, atau klien-server yang tebal. Sistem dengan persyaratan pemeliharaan yang luas mungkin tidak cocok untuk arsitektur client-server berbasis klien atau tebal karena kebutuhan untuk menginstal ulang perangkat lunak pada desktop.

Persyaratan Kinerja. Secara umum, sistem informasi yang memiliki persyaratan kinerja tinggi paling cocok untuk arsitektur client-server. Arsitektur client-server lebih terukur, yang berarti mereka merespon lebih baik terhadap perubahan kebutuhan kapasitas dan dengan demikian memungkinkan organisasi untuk lebih menyesuaikan perangkat keras dengan persyaratan kecepatan sistem. Arsitektur client-server yang memiliki beberapa server di setiap

tingkat harus lebih andal dan memiliki ketersediaan yang lebih besar, karena jika salah satu server mogok, permintaan hanya diteruskan ke server lain, dan pengguna mungkin tidak menyadarinya (walaupun waktu respons bisa lebih buruk). Namun, dalam praktiknya, keandalan dan ketersediaan sangat bergantung pada perangkat keras dan sistem operasi, dan komputer berbasis Windows cenderung memiliki keandalan dan ketersediaan yang lebih rendah daripada komputer Linux atau komputer mainframe.

Persyaratan Keamanan. Secara umum, karena semua perangkat lunak berada di satu lokasi dan karena sistem operasi mainframe lebih aman daripada sistem operasi komputer mikro, arsitektur berbasis server cenderung lebih aman. Untuk alasan ini, sistem bernilai tinggi lebih mungkin ditemukan pada komputer mainframe, bahkan jika mainframe digunakan sebagai server dalam arsitektur client-server. Di dunia yang didominasi Internet saat ini, alat otentikasi dan enkripsi untuk arsitektur client-server berbasis Internet lebih maju daripada arsitektur berbasis server mainframe. Virus adalah masalah potensial di semua arsitektur karena mudah menyebar di komputer desktop. Jika sistem berbasis server dapat mengurangi fungsi yang diperlukan pada sistem desktop, maka mereka mungkin lebih aman.

Persyaratan Budaya dan Politik. Ketika persyaratan budaya dan politik menjadi lebih penting, kemampuan untuk memisahkan logika presentasi dari logika aplikasi dan data menjadi penting. Pemisahan seperti itu memudahkan untuk mengembangkan logika presentasi dalam bahasa yang berbeda sambil menjaga logika aplikasi dan data tetap sama. Ini juga memudahkan untuk menyesuaikan logika presentasi untuk pengguna yang berbeda dan untuk mengubahnya agar lebih memenuhi norma budaya. Sejauh logika presentasi menyediakan akses ke aplikasi dan data, itu juga memudahkan untuk mengimplementasikan versi berbeda yang mengaktifkan atau menonaktifkan fitur berbeda yang diwajibkan oleh undang-undang dan peraturan di berbagai negara. Pemisahan ini adalah yang termudah dalam arsitektur thin client-server, sehingga sistem dengan banyak persyaratan budaya dan politik sering kali menggunakan arsitektur thin client-server. Seperti persyaratan integrasi sistem, dampak persyaratan hukum tergantung pada sifat spesifik persyaratan, tetapi secara umum, sistem berbasis klien cenderung kurang fleksibel.

11.10 VERIFIKASI DAN VALIDASI LAPISAN ARSITEKTUR FISIK

Seperti model pada lapisan lain, desain infrastruktur dan spesifikasi perangkat keras dan perangkat lunak dari lapisan arsitektur fisik perlu diverifikasi dan divalidasi. Memverifikasi dan memvalidasi desain lapisan manajemen data dibagi menjadi tiga kelompok dasar.

Pertama, kami merekomendasikan untuk memverifikasi dan memvalidasi diagram penerapan dengan memastikan bahwa semuanya benar-benar konsisten dan seimbang. Misalnya, setiap node dalam diagram penyebaran model jaringan tingkat atas harus dikaitkan dengan diagram penyebaran terpisah yang mewakili model jaringan tingkat rendah untuk node.

Kedua, spesifikasi perangkat keras dan perangkat lunak harus konsisten dengan model jaringan "tingkat terendah". Misalnya, jika model jaringan tingkat rendah untuk kantor menggambarkan satu set workstation, server, printer, sakelar, router, dll., maka spesifikasi perangkat keras dan perangkat lunak untuk lokasi tersebut harus menjadi detail untuk setiap artefak TI untuk lokasi itu.

Ketiga, setelah sistem diimplementasikan, pengujian persyaratan nonfungsional menjadi sangat penting. Dalam hal ini, pengujian harus dirancang dan dilakukan untuk setiap persyaratan nonfungsional. Misalnya, untuk persyaratan kinerja, pengujian beban harus dilakukan untuk mengidentifikasi kemungkinan kemacetan kinerja dalam jaringan. Kami akan kembali ke topik ini di Bab 12.

Pertanyaan

1. Apa empat fungsi dasar dari setiap sistem informasi?
2. Apa tiga komponen perangkat keras utama dari setiap arsitektur fisik?
3. Sebutkan dua contoh server
4. Bandingkan dan kontraskan arsitektur berbasis server, arsitektur berbasis klien, dan arsitektur berbasis klien-server.
5. Apa masalah terbesar dengan komputasi berbasis server?
6. Apa masalah terbesar dengan komputasi berbasis klien?
7. Jelaskan manfaat utama dan keterbatasan arsitektur thin client-server.
8. Jelaskan manfaat utama dan keterbatasan arsitektur client-server yang tebal.
9. Jelaskan perbedaan antara arsitektur two-tier, threetier, dan n-tier. 10. Tentukan skalabel. Mengapa istilah ini penting bagi pengembang sistem?
10. Apa enam kriteria yang berguna untuk digunakan ketika membandingkan kesesuaian alternatif komputasi?
11. Mengapa tim proyek harus mempertimbangkan arsitektur fisik yang ada dalam organisasi ketika merancang lapisan arsitektur fisik dari sistem baru?
12. Sebutkan tiga jenis awan. Bagaimana mereka berbeda satu sama lain?
13. Apa yang dimaksud dengan arsitektur berorientasi layanan?
14. Definisikan virtualisasi. Bagaimana hubungannya dengan awan?
15. Apa perbedaan antara IaaS, PaaS, dan SaaS?
16. Apa kendala untuk menyediakan lapisan arsitektur fisik dengan teknologi cloud?
17. Jika ada, apa masalah yang terkait dengan keamanan di cloud?
18. Apa itu SOX dan HIPAA, dan bagaimana pengaruhnya terhadap keputusan perusahaan untuk mengadopsi teknologi cloud? 20. Apa yang dimaksud dengan komputasi di mana-mana? Bagaimana dengan Internet of Things?
19. Apa itu objek terpesona? Berikan satu set contoh dari mereka.
20. Apa itu limbah elektronik?
21. Apa masalahnya dengan daur ulang sampah elektronik di halaman belakang?
22. Apa yang dimaksud dengan pusat data hijau?
23. Bagaimana tablet, seperti iPadTM, mengaktifkan kantor tanpa kertas?
24. Biaya tambahan terkait perangkat keras dan perangkat lunak apa yang mungkin perlu disertakan pada spesifikasi perangkat keras dan perangkat lunak?
25. Siapa yang pada akhirnya bertanggung jawab untuk memperoleh perangkat keras dan perangkat lunak untuk suatu proyek?
26. Apa itu tolok ukur, dan mengapa itu penting?
27. Mengapa Hukum Parkinson relevan dengan desain lapisan arsitektur fisik?
28. Menurut Anda apa tiga kesalahan umum yang dilakukan analis pemula dalam desain arsitektur dan spesifikasi perangkat keras dan perangkat lunak?
29. Jelaskan persyaratan nonfungsional utama dan bagaimana mereka mempengaruhi desain lapisan arsitektur fisik.
30. Mengapa berguna untuk mendefinisikan persyaratan nonfungsional secara lebih rinci bahkan jika persyaratan lingkungan teknis menentukan arsitektur tertentu?
31. Apa yang dikomunikasikan model jaringan kepada tim proyek?
32. Apa perbedaan antara model jaringan tingkat atas dan model jaringan tingkat rendah?
33. Apakah beberapa persyaratan nonfungsional lebih penting daripada yang lain dalam mempengaruhi desain arsitektur dan spesifikasi perangkat keras dan perangkat lunak?
34. Menurut Anda apa masalah keamanan paling penting untuk suatu sistem?

Latihan

A. Menggunakan Web (atau majalah industri komputer edisi sebelumnya seperti Computerworld), temukan sistem yang berjalan di lingkungan berbasis server.

Berdasarkan bacaan Anda, menurut Anda mengapa perusahaan memilih lingkungan komputasi itu?

B. Menggunakan Web (atau majalah industri komputer edisi sebelumnya seperti Computerworld), temukan sistem yang berjalan di lingkungan client-server. Berdasarkan bacaan Anda, menurut Anda mengapa perusahaan memilih lingkungan komputasi itu?
C. Menggunakan Web, temukan contoh komponen mainframe, komponen komputer mini, dan komponen komputer mikro. Bandingkan komponen dalam hal harga, kecepatan, memori yang tersedia, dan penyimpanan disk. Apakah Anda menemukan perbedaan harga yang besar ketika kinerja komponen dipertimbangkan?
D. Anda telah dipilih untuk menemukan arsitektur klien-server terbaik untuk sistem entri pesanan berbasis Web yang sedang dikembangkan untuk L.L. Bean. Tulis memo singkat yang menjelaskan kepada manajer proyek alasan Anda memilih arsitektur n-tier daripada arsitektur dua-tier. Dalam memo tersebut, berikan beberapa gagasan tentang komponen berbeda dari arsitektur yang akan Anda sertakan.
E. Pikirkan tentang sistem yang saat ini digunakan universitas Anda untuk layanan karir, dan anggaplah Anda bertanggung jawab untuk mengganti sistem dengan yang baru. Jelaskan bagaimana Anda akan memutuskan arsitektur komputasi untuk sistem baru menggunakan kriteria yang disajikan dalam bab ini. Informasi apa yang perlu Anda temukan sebelum Anda dapat membuat perbandingan yang terdidik dari alternatif-alternatif tersebut?
F. Dengan menggunakan Web, temukan informasi tentang dampak limbah elektronik dan daur ulang halaman belakang di negara berkembang. Berdasarkan temuan Anda, kebijakan TI Hijau apa yang akan Anda sarankan agar diterapkan oleh perusahaan untuk meminimalkan efek negatif limbah elektronik.
G. Menggunakan Web, temukan contoh perusahaan yang mengejar strategi Green IT. Jelaskan apa yang mereka lakukan.
H. Energy Star adalah program bersama antara Departemen Energi AS dan Badan Perlindungan Lingkungan. Apa saja persyaratan agar berbagai perangkat TI disertifikasi sebagai sesuai dengan Energy Star? I. Menggunakan Web, temukan contoh perusahaan yang menggunakan cloud sebagai dasar untuk lapisan arsitektur fisik. Jelaskan dengan tepat apa yang mereka lakukan.
I. Temukan perusahaan produk konsumen di Web dan baca deskripsi perusahaannya (sehingga Anda mendapatkan pemahaman yang baik tentang lokasi geografis perusahaan). Berpura-puralah bahwa perusahaan akan membuat aplikasi baru untuk mendukung penjualan eceran melalui Web. Buat model jaringan tingkat tinggi yang menggambarkan lokasi yang akan menyertakan komponen yang mendukung aplikasi ini.
J. Buat diagram jaringan tingkat rendah untuk gedung yang menampung laboratorium komputer di universitas Anda. Pilih aplikasi (misalnya, pendaftaran kursus, penerimaan siswa) dan sertakan hanya komponen yang relevan dengan aplikasi itu.
K. Sebuah perusahaan semen dengan kantor pusat di Gresik, Jawa Timur, sedang mempertimbangkan untuk mengembangkan sistem untuk melacak efisiensi Semennya di Provinsi luar Jawa Timur. Setiap minggu, dari ratusan kg semen akan mengunggah data kinerja melalui satelit ke mainframe perusahaan di Tuban. Manajer produksi di setiap lokasi akan menggunakan komputer pribadi untuk terhubung ke penyedia layanan Internet dan mengakses laporan melalui Web. Buat model jaringan tingkat tinggi yang menggambarkan lokasi yang memiliki komponen pendukung sistem ini.
L. Misalkan ibu Anda adalah seorang agen real estat, dan dia telah memutuskan untuk mengotomatiskan tugas sehari-harinya menggunakan komputer laptop. Pertimbangkan potensi kebutuhan perangkat keras dan perangkat lunaknya, dan buat spesifikasi perangkat keras dan perangkat lunak yang menjelaskannya. Spesifikasi

harus dikembangkan untuk membantu ibu Anda membeli perangkat keras dan perangkat lunaknya sendiri.

M. Misalkan kantor penerimaan di universitas Anda memiliki aplikasi berbasis Web sehingga siswa dapat mendaftar untuk masuk secara online. Baru-baru ini, ada dorongan untuk menerima lebih banyak siswa internasional ke universitas. Apa yang Anda rekomendasikan agar aplikasi disertakan untuk memastikan bahwa itu mendukung persyaratan global ini?
N. Berdasarkan masalah A Real Estate Inc. di Bab 4 (latihan I, J, dan K), Bab 5 (latihan P dan Q), Bab 6 (latihan D), Bab 7 (latihan A), Bab 8 (latihan A), Bab 9 (latihan L), dan Bab 10 (latihan I dan J), menyarankan desain arsitektur fisik dan menggambarkannya dengan diagram penerapan.
O. Berdasarkan masalah A Video Store di Bab 4 (latihan L, M, dan N), Bab 5 (latihan R dan S), Bab 6 (latihan E), Bab 7 (latihan B), Bab 8 (latihan B), Bab 9 (latihan M), dan Bab 10 (latihan K dan L), menyarankan desain arsitektur fisik dan menggambarkannya dengan diagram penerapan.
P. Berdasarkan masalah keanggotaan gym pada Bab 4 (latihan O, P, dan Q), Bab 5 (latihan T dan U), Bab 6 (latihan F), Bab 7 (latihan C), Bab 8 (latihan C), Bab 9 (latihan N), dan Bab 10 (latihan M dan N), menyarankan desain arsitektur fisik dan menggambarkannya dengan diagram penerapan.
Q.
R. Berdasarkan latihan Piknik R Us di Bab 4 (latihan R, S, dan T), Bab 5 (latihan V dan W), Bab 6 (latihan G), Bab 7 (latihan D), Bab 8 (latihan D), dan Bab 9 (latihan O), dan Bab 10 (latihan O dan P), menyarankan desain arsitektur fisik dan menggambarkannya dengan diagram penerapan.
S. Berdasarkan masalah Klub Bulanan di Bab 4 (latihan U, V, dan W), Bab 5 (latihan X dan Y), Bab 6 (latihan H), Bab 7 (latihan E), Bab 8 (latihan E), Bab 9 (latihan N), dan Bab 10 (latihan Q dan R), menyarankan desain arsitektur fisik dan menggambarkannya dengan diagram penyebaran.

Minicase

1. Tim proyek pengembangan sistem di sekolah SMK Bahasa telah bekerja untuk mendefinisikan desain arsitektur fisik untuk sistem. Fokus utama dari proyek ini adalah sistem operasi lokasi sekolah berjaringan, yang memungkinkan setiap lokasi sekolah untuk dengan mudah merekam dan mengambil semua data transaksi lokasi sekolah. Elemen sistem lainnya adalah penggunaan Internet untuk memungkinkan siswa saat ini dan calon siswa melihat penawaran kelas di salah satu lokasi, menjadwalkan pelajaran dan mendaftar di kelas di lokasi yg ditentukan, dan mempertahankan profil kemajuan siswa—a analisis rahasia pengembangan keterampilan bahasa siswa.

   Tim proyek telah mempertimbangkan isu globalisasi yang harus diperhitungkan dalam desain arsitektur. Rencana sekolah untuk ekspansi ke Jepang terus berlanjut. Lokasi sekolah Jepang pertama secara tentatif direncanakan untuk dibuka sekitar enam bulan setelah data target penyelesaian untuk proyek sistem. Oleh karena itu, penting bahwa isu-isu yang berkaitan dengan lokasi internasional ditangani sekarang selama desain.

   Asumsikan bahwa Anda telah diberi tanggung jawab untuk menyiapkan ringkasan memo tentang isu-isu globalisasi yang harus diperhitungkan dalam desain. Siapkan memo ini yang membahas isu globalisasi yang relevan dengan sistem baru SMK Bahasa.

2. Jenni adalah anggota tim proyek yang relatif baru yang mengembangkan sistem manajemen toko ritel untuk rantai toko barang olahraga. Kantor pusat perusahaan berada di Jakarta, dan jaringan tersebut memiliki dua puluh tujuh lokasi di seluruh Bandung, Semarang dan Surabaya. Beberapa kota memiliki banyak toko.

   Sistem baru akan menjadi arsitektur client-server jaringan. Toko akan ditautkan ke salah satu dari tiga server regional, dan server regional akan ditautkan ke kantor pusat perusahaan di Semarang. Server regional juga terhubung satu sama lain. Setiap toko ritel akan dilengkapi dengan konfigurasi serupa dari dua terminal point-of-sale berbasis PC yang terhubung ke server file lokal. Jenni telah diberi tugas untuk mengembangkan model jaringan yang akan mendokumentasikan struktur geografis dari sistem ini. Dia belum pernah menghadapi sistem lingkup ini sebelumnya dan sedikit tidak yakin bagaimana memulainya.

   a. Siapkan satu set instruksi untuk diikuti Jenni dalam mengembangkan model jaringan ini.
   b. Menggunakan diagram penerapan, gambar model jaringan untuk organisasi ini.
   c. Siapkan satu set instruksi untuk diikuti Jenni dalam mengembangkan spesifikasi perangkat keras dan perangkat lunak.
3. Lihat minicase Professional and Scientific Staff Management (PSSM) di Bab 4, 6, 7, 8, 9, dan 10. Berdasarkan solusi yang dikembangkan untuk masalah tersebut, sarankan desain arsitektur fisik dan gambarkan dengan diagram penerapan.
4. Lihat minicase Holiday Travel Vehicles di Bab 5, 6, 7, 8, 9, dan 10. Berdasarkan solusi yang dikembangkan untuk masalah tersebut, sarankan desain arsitektur fisik dan gambarkan dengan diagram penerapan.

BAGIAN TIGA

KONSTRUKSI, INSTALASI DAN OPERASI

Selama konstruksi, sistem yang sebenarnya dibangun. Membangun sistem informasi yang sukses memerlukan serangkaian aktivitas: pemrograman, pengujian, dan pendokumentasian sistem. Dalam perekonomian global saat ini, isu budaya juga memegang peranan penting dalam mengelola kegiatan tersebut. Instalasi sistem informasi memerlukan peralihan dari sistem saat ini ke sistem baru. Proses konversi ini bisa sangat terlibat; misalnya, perbedaan budaya di antara pengguna, tim pengembangan, dan kedua kelompok bisa sangat menantang. Selain itu, konversi tidak hanya melibatkan mematikan sistem lama dan menyalakan yang baru, tetapi juga dapat melibatkan upaya pelatihan yang signifikan. Akhirnya, mengoperasikan sistem dapat mengungkap persyaratan tambahan yang mungkin harus ditangani oleh tim pengembangan.

BAB 12

KONSTRUKSI

Bab ini membahas aktivitas yang diperlukan untuk berhasil membangun sistem informasi: pemrograman, pengujian, dan pendokumentasian sistem. Pemrograman memakan waktu dan mahal, tetapi kecuali dalam keadaan yang tidak biasa, ini adalah yang paling sederhana untuk analis sistem karena dipahami dengan baik. Untuk alasan ini, analis sistem berfokus pada pengujian (membuktikan bahwa sistem bekerja seperti yang dirancang) dan mengembangkan dokumentasi.

12.1 Tujuan

- Memahami isu-isu dasar yang berkaitan dengan mengelola programmer.
- Pahami bagaimana masalah budaya dapat memengaruhi efisiensi, efektivitas, dan fokus tim pengembangan perangkat lunak.
- Kenali berbagai jenis dokumentasi.
- Memahami cara mengembangkan dokumentasi.
- Memahami bagaimana orientasi objek mempengaruhi pengujian perangkat lunak.
- Memahami berbagai jenis dan tujuan unit test.
- Memahami berbagai jenis dan tujuan tes integrasi.
- Memahami berbagai jenis dan tujuan pengujian sistem.
- Memahami berbagai jenis dan tujuan tes penerimaan.

12.2 Pendahuluan

Ketika orang pertama kali belajar tentang mengembangkan sistem informasi, mereka biasanya langsung berpikir tentang menulis program. Pemrograman dapat menjadi komponen tunggal terbesar dari setiap proyek pengembangan sistem dalam hal waktu dan biaya. Namun, itu juga bisa menjadi komponen yang paling dipahami dan oleh karena itu—kecuali dalam keadaan yang jarang terjadi—menawarkan masalah paling sedikit dari semua aspek pengembangan sistem. Ketika proyek gagal, biasanya bukan karena pemrogram tidak dapat menulis program, tetapi karena analisis, desain, instalasi, dan/atau manajemen proyek dilakukan dengan buruk.

Konstruksi adalah pengembangan semua bagian dari sistem, termasuk perangkat lunak itu sendiri, dokumentasi, dan prosedur operasi baru. Melihat kembali Gambar 1-18, kita melihat bahwa fase Konstruksi dari Proses Terpadu yang Disempurnakan terutama berkaitan dengan alur kerja Implementasi, Pengujian, dan Konfigurasi dan Manajemen Perubahan. Implementasi jelas berhubungan dengan pemrograman. Pemrograman sering dilihat sebagai titik fokus pengembangan sistem. Bagaimanapun, pengembangan sistem adalah menulis program. Ini adalah alasan kami melakukan semua analisis dan desain. Dan itu menyenangkan. Banyak pemrogram pemula melihat pengujian dan dokumentasi sebagai hal yang mengganggu setelah beberapa pemikiran. Pengujian dan dokumentasi tidak menyenangkan, sehingga sering kurang mendapat perhatian daripada aktivitas kreatif menulis program.

Namun, pemrograman dan pengujian sangat mirip dengan menulis dan mengedit. Tidak ada penulis profesional (atau siswa yang baik yang menulis makalah penting) akan berhenti setelah menulis draf pertama. Membaca ulang, mengedit, dan merevisi draf awal menjadi karya tulis yang baik adalah ciri-ciri tulisan yang baik. Demikian juga, pengujian menyeluruh adalah ciri khas pengembang perangkat lunak profesional. Sebagian besar organisasi profesional mencurahkan lebih banyak waktu dan uang untuk pengujian (dan revisi berikutnya dan pengujian ulang) daripada menulis program di tempat pertama.

Alasannya adalah ekonomi sederhana: Waktu henti dan kegagalan yang disebabkan oleh bug perangkat lunak sangat mahal. Banyak organisasi besar memperkirakan biaya downtime aplikasi penting sebesar Rp. 7.500.000 hingga Rp. 15.000.000 per jam. Satu bug serius yang menyebabkan satu jam downtime dapat menghabiskan lebih dari satu tahun gaji programmer—dan seberapa sering bug ditemukan dan diperbaiki dalam satu jam ? Oleh karena itu, pengujian adalah bentuk asuransi. Organisasi bersedia menghabiskan banyak waktu dan uang untuk mencegah kemungkinan kegagalan besar setelah sistem diinstal.

Oleh karena itu, sebuah program biasanya tidak dianggap selesai sampai tes untuk program tersebut lulus. Untuk alasan ini, pemrograman dan pengujian sangat erat, dan karena pemrograman adalah pekerjaan utama programmer (bukan analis), pengujian (bukan pemrograman) sering menjadi fokus tahap konstruksi untuk tim analisis sistem.

Alur kerja Manajemen Konfigurasi dan Perubahan melacak status sistem yang berkembang. Sistem informasi yang berkembang terdiri dari seperangkat artefak yang mencakup, misalnya, diagram, kode sumber, dan yang dapat dieksekusi. Selama proses pengembangan, artefak ini dimodifikasi. Jumlah pekerjaan, dan karenanya dolar, yang digunakan untuk pengembangan artefak sangat besar. Oleh karena itu, artefak itu sendiri harus ditangani seperti halnya aset yang mahal akan ditangani: Kontrol akses harus diterapkan untuk melindungi artefak agar tidak dicuri atau dihancurkan. Karena artefak dimodifikasi secara teratur, jika tidak secara terus menerus, mekanisme kontrol versi yang baik harus dibuat. Ketertelusuran artefak kembali melalui berbagai artefak yang dikembangkan, seperti desain lapisan manajemen data, diagram kelas, diagram paket, dan diagram Use-Case, dengan persyaratan khusus juga sangat penting. Tanpa ketertelusuran ini, kita tidak akan tahu aspek mana dari sistem yang harus dimodifikasi ketika—bukan jika—persyaratannya berubah.

12.3 MENGELOLA PEMROGRAMAN

Secara umum, analis sistem tidak menulis program; programmer menulis program. Oleh karena itu, tugas utama analis sistem selama pemrograman adalah . . . menunggu. Namun, manajer proyek biasanya sangat sibuk mengelola upaya pemrograman dengan menugaskan programmer, mengkoordinasikan kegiatan, dan mengelola jadwal pemrograman.

Menugaskan Pemrogram

Langkah pertama dalam pemrograman adalah menugaskan modul ke programmer. Seperti yang dibahas dalam Bab 8, setiap modul (kelas, objek, atau metode) harus terpisah dan berbeda dari modul lain (yaitu, kohesi harus dimaksimalkan dan coupling harus diminimalkan). Manajer proyek pertama-tama mengelompokkan kelas-kelas yang terkait sehingga setiap programmer bekerja pada kelas-kelas terkait. Kelompok kelas ini kemudian ditugaskan ke pemrogram. Tempat yang baik untuk memulai adalah dengan melihat diagram paket.

Salah satu aturan pengembangan sistem adalah semakin banyak programmer yang terlibat dalam suatu proyek, semakin lama waktu yang dibutuhkan untuk membangun sistem. Ini karena dengan bertambahnya ukuran tim pemrograman, kebutuhan akan koordinasi meningkat secara eksponensial, dan semakin banyak koordinasi yang diperlukan, semakin sedikit waktu yang dapat dihabiskan programmer untuk menulis sistem (lihat Bab 2). Ukuran terbaik adalah tim pemrograman sekecil mungkin. Ketika proyek sangat kompleks sehingga membutuhkan tim yang besar, strategi terbaik adalah mencoba memecah proyek menjadi serangkaian bagian yang lebih kecil yang dapat berfungsi semandiri mungkin.

Kegiatan Koordinasi

Koordinasi dapat dilakukan melalui sarana berteknologi tinggi dan berteknologi rendah. Pendekatan paling sederhana adalah mengadakan pertemuan proyek mingguan untuk membahas perubahan apa pun pada sistem yang muncul selama seminggu terakhir—atau masalah apa pun yang muncul. Ingat, pendekatan pengembangan tangkas seperti Scrum mendorong pertemuan harian (lihat Bab 2). Pertemuan rutin, meskipun singkat, mendorong meluasnya komunikasi dan diskusi masalah sebelum menjadi masalah.

Cara penting lainnya untuk meningkatkan koordinasi adalah dengan membuat dan mengikuti standar yang dapat berkisar dari aturan formal untuk penamaan file, hingga formulir yang harus dilengkapi ketika tujuan tercapai, hingga pedoman pemrograman (lihat Bab 2). Ketika sebuah tim membentuk standar dan kemudian mengikutinya, proyek dapat diselesaikan lebih cepat karena koordinasi tugas tidak terlalu rumit.

Analis juga harus menerapkan mekanisme untuk menjaga agar upaya pemrograman tetap terorganisir dengan baik. Banyak tim proyek menyiapkan tiga area di mana programmer dapat bekerja: area pengembangan, area pengujian, dan area produksi. Area ini dapat berupa direktori yang berbeda pada hard disk server, server yang berbeda, atau lokasi fisik yang berbeda, tetapi intinya adalah file, data, dan program dipisahkan berdasarkan status penyelesaiannya. Pada awalnya, pemrogram mengakses dan membangun file di dalam area pengembangan dan kemudian menyalinnya ke area pengujian ketika pemrogram selesai. Jika sebuah program tidak lulus tes, itu dikirim kembali ke pengembangan. Setelah semua program diuji dan siap untuk mendukung sistem baru, program tersebut disalin ke area produksi—lokasi di mana sistem final akan berada.

Menyimpan file dan program di tempat yang berbeda berdasarkan status penyelesaian membantu mengelola kontrol perubahan, tindakan mengoordinasikan sistem saat sistem berubah melalui konstruksi. Teknik kontrol perubahan lainnya adalah melacak pemrogram mana yang mengubah kelas dan paket mana dengan menggunakan log program. Log hanyalah sebuah formulir di mana pemrogram keluar dari kelas dan paket untuk menulis dan masuk ketika mereka selesai. Baik area pemrograman dan log program membantu analis memahami dengan tepat siapa yang telah mengerjakan apa dan status sistem saat ini. Tanpa teknik ini, file dapat dimasukkan ke dalam produksi tanpa pengujian yang tepat (misalnya, dua programmer dapat mulai bekerja pada kelas atau paket yang sama pada waktu yang sama).

Jika CASE tool digunakan selama langkah konstruksi, itu bisa sangat membantu untuk kontrol perubahan karena banyak CASE tool diatur untuk melacak status program dan membantu mengelola pemrogram saat mereka bekerja. Dalam kebanyakan kasus, menjaga koordinasi tidak rumit secara konseptual. Itu hanya membutuhkan banyak disiplin dan perhatian untuk melacak detail kecil.

Mengelola Jadwal

Perkiraan waktu yang dihasilkan selama identifikasi proyek dan disempurnakan selama analisis dan desain hampir selalu perlu disempurnakan saat proyek berlangsung selama konstruksi karena hampir tidak mungkin untuk mengembangkan penilaian yang tepat dari jadwal proyek. Seperti yang telah kita bahas di Bab 2, satu set estimasi waktu yang dilakukan dengan baik biasanya memiliki margin kesalahan 10 persen pada saat kita mencapai langkah konstruksi. Sangat penting bahwa perkiraan waktu direvisi saat konstruksi berlangsung. Jika modul program membutuhkan waktu lebih lama untuk dikembangkan dari yang diharapkan, maka respons yang bijaksana adalah memindahkan tanggal penyelesaian yang diharapkan nanti dengan jumlah yang sama.

Salah satu penyebab paling umum untuk masalah jadwal adalah scope creep. Scope creep terjadi ketika persyaratan baru ditambahkan ke proyek setelah desain sistem diselesaikan (lihat Bab 2). Jika Anda ingat, Scrum mendorong penambahan persyaratan baru ke backlog produk alih-alih membiarkan ruang lingkup proyek berubah di antara sprint. Scope creep bisa sangat mahal karena perubahan yang dibuat di akhir pengembangan sistem dapat membutuhkan banyak desain sistem yang telah selesai (dan bahkan program yang sudah ditulis) untuk dikerjakan ulang. Setiap perubahan yang diusulkan selama konstruksi harus memerlukan persetujuan dari manajer proyek dan hanya boleh dilakukan setelah analisis biaya-manfaat yang cepat dilakukan.

Penyebab umum lainnya adalah slippage hari demi hari yang tidak diketahui dalam jadwal. Satu paket terlambat sehari di sini; satu lagi adalah satu hari terlambat di sana. Tak lama kemudian penundaan kecil ini bertambah dan proyek terasa terlambat dari jadwal. Sekali lagi, utama untuk mengelola upaya pemrograman adalah dengan memperhatikan kesalahan kecil ini dengan hati-hati dan memperbarui jadwal yang sesuai.

Biasanya, seorang manajer proyek membuat penilaian risiko yang melacak potensi risiko bersama dengan evaluasi kemungkinan dan potensi dampaknya. Saat langkah konstruksi mendekati penutupan, daftar risiko berubah karena beberapa item dihapus dan yang lain muncul. Manajer proyek terbaik, bagaimanapun, bekerja keras untuk menjaga risiko agar tidak berdampak pada jadwal dan biaya yang terkait dengan proyek.

Menghindari Kesalahan Implementasi Klasik

Dalam bab-bab sebelumnya, kita telah membahas kesalahan klasik dan bagaimana menghindarinya. Di sini, kami merangkum empat kesalahan klasik dalam implementasi:

a. *Pengembangan berorientasi penelitian*: Menggunakan teknologi mutakhir membutuhkan pengembangan berorientasi penelitian yang mengeksplorasi teknologi baru karena alat dan teknik "peningkatan" tidak dipahami dengan baik, tidak didokumentasikan dengan baik, dan tidak berfungsi persis seperti yang dijanjikan.

   Solusi: Jika Anda menggunakan teknologi tercanggih, Anda perlu meningkatkan perkiraan waktu dan biaya proyek secara signifikan bahkan jika (beberapa ahli akan

mengatakan terutama jika) teknologi tersebut mengklaim dapat mengurangi waktu dan tenaga.

b. *Menggunakan personel berbiaya rendah*: Anda mendapatkan apa yang Anda bayar. Konsultan atau staf dengan biaya terendah secara signifikan kurang produktif daripada staf terbaik. Beberapa penelitian telah menunjukkan bahwa pemrogram terbaik menghasilkan perangkat lunak enam hingga delapan kali lebih cepat daripada yang paling tidak produktif (namun biayanya hanya 50 hingga 100 persen lebih mahal).

   Solusi: Jika biaya merupakan masalah kritis, tetapkan personel yang terbaik dan paling mahal; jangan pernah menugaskan personel tingkat pemula dalam upaya menghemat biaya.

c. *Kurangnya kontrol kode*: Pada proyek besar, pemrogram perlu mengoordinasikan perubahan pada kode sumber program (sehingga dua pemrogram tidak mencoba mengubah program yang sama pada saat yang sama dan menimpa perubahan satu sama lain). Meskipun prosedur manual tampaknya berhasil (misalnya, mengirim catatan email kepada orang lain saat Anda mengerjakan sebuah program untuk memberi tahu mereka agar tidak melakukannya), kesalahan tidak dapat dihindari.

   Solusi: Gunakan pustaka kode sumber yang mengharuskan pemrogram untuk "memeriksa" program dan melarang orang lain mengerjakannya secara bersamaan.

d. *Pengujian yang tidak memadai:* Alasan nomor satu untuk kegagalan proyek selama implementasi adalah pengujian ad hoc—di mana programmer dan analis menguji sistem tanpa rencana pengujian formal.

   Solusi: Selalu alokasikan waktu yang cukup dalam rencana proyek untuk pengujian formal.

Masalah Budaya

Salah satu masalah utama yang dihadapi organisasi pengembangan sistem informasi adalah aspek implementasi pengembangan sistem informasi. Konflik yang disebabkan oleh perbedaan budaya nasional dan organisasi kini menjadi perhatian nyata. Dengan potensi Komputasi Cloud (lihat Bab 11) yang berpotensi memungkinkan lebih banyak outsourcing, potensi konflik budaya bahkan lebih besar.

Contoh sederhana yang dapat menunjukkan perbedaan budaya dalam pembelajaran siswa adalah gagasan plagiarisme. Apa sebenarnya yang tersirat dari plagiarisme? Budaya yang berbeda memiliki pandangan yang sangat berbeda. Dalam beberapa budaya, salah satu bentuk penghormatan tertinggi adalah dengan mengutip seorang ahli. Namun, dalam budaya yang sama ini, tidak perlu merujuk ahlinya. Tindakan mengutip ahli itu sendiri adalah tindakan menghormati. Dalam beberapa kasus, sebenarnya merujuk pakar melalui penggunaan tanda kutip dan catatan kaki dapat dipandang sebagai penghinaan bagi pakar dan pembaca karena jelas bagi pembaca bahwa penulis tidak mengharapkan pembaca untuk mengenali kutipan pakar. Harapan ini disebabkan oleh ketidaktahuan pembaca sendiri atau kurangnya reputasi pakar. Either way, penulis akan menghina seseorang melalui penggunaan tanda kutip dan catatan kaki. Budaya ini cenderung bersifat kolektivis (lihat Bab 10 dan 13). Akibatnya, karena kolektif memiliki semua ide, tidak ada konsep pencurian ide. Namun, di Indonesia, yang terjadi justru sebaliknya. Jika seorang penulis tidak menggunakan tanda kutip dan catatan kaki untuk memberikan kredit dengan tepat ke sumber kutipan (atau parafrase), maka

penulis bersalah atas pencurian (sebut saja plagiasi/plagiarisme). Jelas, di dunia global saat ini, plagiarisme bukanlah masalah sederhana.

Contoh sederhana lain dari perbedaan budaya, berkaitan dengan pembelajaran siswa, adalah gagasan siswa bekerja sama untuk menyelesaikan tugas pekerjaan rumah. Meskipun kita semua tahu bahwa penelitian telah menunjukkan bahwa siswa belajar lebih baik dalam kelompok, di Indonesia, kami melihat siswa yang menyerahkan tugas karangan bahasa indonesia dengan jawaban sama persis maka dianggap menyontek. Dalam budaya lain, kinerja individu tidak sepenting kinerja kelompok. Sekali lagi, budaya ini bersifat kolektivis. Akibatnya, membantu sesama siswa untuk memahami tugas dan untuk tampil lebih baik di kelas akan menjadi harapan. Selanjutnya, sikap ini meluas ke pengambilan tes. Jika seorang siswa sedang berjuang dalam ujian dan jika Anda berasal dari budaya kolektivis, adalah tugas Anda untuk mengizinkan sesama siswa Anda menyalin jawaban Anda. Jelas, ini adalah contoh lain dari perbedaan budaya yang substantif. Dari perspektif bisnis, perbedaan pandangan tentang plagiarisme dan kecurangan ini dapat memiliki implikasi serius terhadap perlindungan kekayaan intelektual.

Seperti yang kami nyatakan sebelumnya, dengan outsourcing lepas pantai, tim pengembangan sistem informasi dapat tersebar secara geografis dan multikultural dalam keanggotaan mereka. Mengingat masalah di atas dan ketika kita mempertimbangkan perbedaan budaya yang diidentifikasi Hall dan Hofstede (lihat Bab 10), masalah budaya menambah kerutan baru dalam pengelolaan pengembangan sistem informasi yang sukses. Dari perspektif pengembangan sistem informasi, konteks dapat memengaruhi kemampuan anggota tim untuk melihat (atau tidak melihat) solusi kreatif potensial yang out of the box atau memengaruhi kemampuan (atau ketidakmampuan) anggota tim untuk memahami seluruh masalah yang sedang dipertimbangkan. Lebih jauh lagi, mengingat dimensi ini, tingkat detail arah dapat bervariasi antar budaya. Dimensi individualisme dan kolektivisme Hofstede sebagian menjelaskan hasil mengenai plagiarisme dan kecurangan yang dijelaskan di atas. Mengingat pentingnya kekayaan intelektual dalam TI, hal ini berpotensi menjadi masalah nyata ketika melepaskan pengembangan ke budaya kolektivis. Selain itu, kecepatan pesan dan dimensi konteks Hall juga dapat memengaruhi cara menangani hal ini. Tergantung pada budayanya, terlalu banyak detail bisa menghina, tetapi mencoba menempatkan masalah ini ke dalam bingkai kontekstual yang sensitif secara budaya itu sulit.

Saat mengelola programmer dalam pengaturan multikultural, dimensi waktu Hall juga harus dipertimbangkan. Dalam budaya waktu monokronis, tenggat waktu sangat penting. Ini mungkin mengapa timeboxing relatif berhasil sebagai metode untuk mengontrol proyek (lihat Bab 2). Namun, dalam budaya waktu polikronis, tenggat waktu tidak lebih dari sekadar saran. Jelas, ketika mengelola programmer, memahami bagaimana budaya mempertimbangkan waktu sangat penting untuk memiliki pengiriman produk yang sukses dan proses pengembangan yang sukses.

Dimensi Hofstede lainnya yang disebutkan sebelumnya adalah jarak kekuasaan, penghindaran ketidakpastian, dan maskulinitas versus feminitas. Mengelola programmer dalam budaya dengan nilai jarak daya tinggi berbeda dengan budaya dengan jarak daya rendah. Misalnya, di Indonesia, programmer melihat diri mereka setara dengan manajer mereka. Bahkan, di beberapa perusahaan, presiden perusahaan dapat menemukan solusi "pengkodean" di samping karyawan baru. Ini agak menjelaskan semakin populernya metode tangkas (lihat Bab 1). Sebagai perbandingan, dalam budaya jarak kekuasaan yang tinggi, presiden perusahaan tidak akan pernah menyerah untuk melakukan tugas yang sama seperti karyawan baru. Itu akan menghina presiden dan mempermalukan pegawai baru.

Berkenaan dengan penghindaran ketidakpastian, pilihan pendekatan pengembangan sistem dapat terpengaruh. Dalam budaya yang lebih suka segala sesuatunya rapi dan teratur, metodologi pengembangan sistem yang sangat berorientasi pada aturan akan bermanfaat. Juga, sertifikasi profesional anggota tim pengembangan dan sertifikasi tim dan perusahaan ISO atau CMMI akan memberikan kredibilitas kepada tim, sedangkan dalam budaya yang rela mengambil risiko, sertifikasi mungkin tidak meningkatkan persepsi posisi tim pengembangan.

Saat mengelola programmer dalam budaya maskulin, sangat penting untuk memberikan pengakuan kepada anggota tim pengembangan yang berkinerja terbaik dan juga untuk mengenali tim dengan kinerja terbaik. Di sisi lain, ketika mempertimbangkan budaya feminin, lebih penting untuk memastikan bahwa tempat kerja adalah lingkungan yang mendukung, tidak kompetitif, dan memelihara.

Hofstede telah mengidentifikasi dimensi kelima, orientasi jangka panjang versus jangka pendek, yang berhubungan dengan bagaimana budaya memandang masa lalu dan masa depan. Dalam budaya fokus jangka panjang, pengembangan tim dan hubungan yang mendalam dengan klien sangat penting, sedangkan dalam budaya yang menekankan jangka pendek, memberikan produk berkualitas tinggi tepat waktu adalah yang terpenting.

Selama bertahun-tahun, manajer proyek di Indonesia harus mengumpulkan individu dari latar belakang yang sangat berbeda. Selain itu, selalu ada bahasa lisan dan tulisan yang sama, bahasa Inggris, dan ide peleburan yang menjamin beberapa tingkat kesamaan di antara anggota tim. Namun, di "dunia datar" dewasa ini, tidak ada lagi budaya yang sama atau bahasa lisan dan tulisan yang sama. Dari perspektif pengembangan sistem informasi, bahasa umum cenderung UML, Java, SQL, C++, Objective-C, dan Visual Basic, bukan bahasa Inggris. Namun, saat ini, tidak ada budaya umum. Akibatnya, memahami isu-isu budaya akan menjadi sangat penting dalam waktu dekat untuk berhasil mengelola tim pengembangan internasional dan multikultural.

12.4 MENGEMBANGKAN DOKUMENTASI

Pengembangan dokumentasi sistem harus dilakukan selama pengembangan sistem. Dalam banyak hal, dokumentasi sebuah sistem adalah sebuah sistem. Jadi, mengembangkan dokumentasi dapat mengikuti pendekatan yang serupa, tetapi lebih sederhana, seperti pengembangan perangkat lunak. Dalam hal ini, membuat Use-Case dan mengembangkan antarmuka pengguna ke dokumentasi masuk akal. Ada dua jenis dokumentasi yang berbeda secara mendasar: dokumentasi sistem dan dokumentasi pengguna. Dokumentasi sistem dimaksudkan untuk membantu pemrogram dan analis sistem memahami perangkat lunak aplikasi dan memungkinkan mereka untuk membangun atau memeliharanya setelah sistem diinstal. Dokumentasi sistem sebagian besar merupakan produk sampingan dari analisis sistem dan proses desain dan dibuat saat proyek dibuka. Setiap langkah dan fase menghasilkan dokumen yang penting dalam memahami bagaimana sistem dibangun atau akan dibangun, dan dokumen-dokumen ini disimpan dalam pengikat proyek. Di banyak lingkungan pengembangan berorientasi objek, adalah mungkin untuk mengotomatisasi pembuatan dokumentasi terperinci untuk kelas dan metode. Misalnya, di Java, jika pemrogram menggunakan komentar bergaya javadoc, maka dimungkinkan untuk membuat halaman HTML yang mendokumentasikan kelas dan metodenya secara otomatis dengan menggunakan utilitas javadoc. Karena kebanyakan programmer melihat dokumentasi dengan sangat tidak suka, apapun yang dapat membuat dokumentasi lebih mudah dibuat akan berguna.

Dokumentasi pengguna (seperti manual pengguna, manual pelatihan, dan sistem bantuan online) dirancang untuk membantu pengguna mengoperasikan sistem. Meskipun sebagian besar tim proyek mengharapkan pengguna telah menerima pelatihan dan telah membaca manual pengguna sebelum mengoperasikan sistem, sayangnya, hal ini tidak selalu

terjadi. Saat ini lebih umum—terutama dalam kasus paket perangkat lunak komersial untuk mikrokomputer—pengguna mulai menggunakan perangkat lunak tanpa pelatihan atau membaca manual pengguna. Di bagian ini, kami fokus pada dokumentasi pengguna.

Dokumentasi pengguna sering dibiarkan sampai akhir proyek, yang merupakan strategi berbahaya. Mengembangkan dokumentasi yang baik membutuhkan waktu lebih lama daripada yang diperkirakan banyak orang karena membutuhkan lebih dari sekadar menulis beberapa halaman. Memproduksi dokumentasi memerlukan perancangan dokumen (baik di atas kertas atau online), menulis teks, mengedit dokumen, dan mengujinya. Untuk dokumentasi berkualitas baik, proses ini biasanya memakan waktu sekitar tiga jam per halaman (spasi tunggal) untuk dokumentasi berbasis kertas atau dua jam per layar untuk dokumentasi online. Jadi dokumentasi "sederhana", seperti manual pengguna sepuluh halaman dan satu set dua puluh layar bantuan, membutuhkan waktu tujuh puluh jam. Tentu saja, dokumentasi berkualitas rendah dapat dihasilkan lebih cepat.

Waktu yang dibutuhkan untuk mengembangkan dan menguji dokumentasi pengguna harus dimasukkan ke dalam rencana proyek. Sebagian besar organisasi berencana untuk memulai pengembangan dokumentasi setelah desain antarmuka dan spesifikasi program selesai. Draf awal dokumentasi biasanya dijadwalkan untuk diselesaikan segera setelah unit test selesai. Ini mengurangi (tetapi tidak menghilangkan) kemungkinan bahwa dokumentasi perlu diubah karena perubahan perangkat lunak dan masih menyisakan cukup waktu untuk dokumentasi diuji dan direvisi sebelum tes penerimaan dimulai.

Meskipun manual berbasis kertas masih penting, dokumentasi online menjadi lebih meresap. Dokumentasi berbasis kertas lebih mudah digunakan karena lebih familiar bagi pengguna, terutama pemula yang kurang memiliki pengalaman komputer; dokumentasi online mengharuskan pengguna untuk mempelajari satu set perintah lagi. Dokumentasi berbasis kertas juga lebih mudah untuk dibolak-balik dan mendapatkan pemahaman umum tentang organisasi dan topiknya dan dapat digunakan jauh dari komputer itu sendiri.

Ada empat kekuatan utama dari dokumentasi online yang semuanya menjamin bahwa itu akan menjadi bentuk dominan untuk abad ke-21. Pencarian informasi seringkali lebih sederhana (asalkan indeks pencarian bantuan dirancang dengan baik) karena pengguna dapat mengetikkan berbagai kata utama untuk melihat informasi hampir secara instan, daripada harus mencari melalui indeks atau daftar isi dalam dokumen kertas. Informasi yang sama dapat disajikan beberapa kali dalam berbagai format, sehingga pengguna dapat menemukan dan membaca informasi dengan cara yang paling informatif (redundansi seperti itu dimungkinkan dalam dokumentasi kertas, tetapi biaya dan ukuran manual yang dihasilkan membuatnya tidak praktis.). Dokumentasi online menyediakan banyak cara baru bagi pengguna untuk berinteraksi dengan dokumentasi yang tidak mungkin dilakukan dalam dokumentasi kertas statis. Misalnya, dimungkinkan untuk menggunakan tautan atau "tips alat" (yaitu, teks pop-up; lihat Bab 10) untuk menjelaskan istilah yang tidak dikenal, dan seseorang dapat menulis rutinitas "tampilkan-saya" yang menunjukkan di layar tombol apa klik dan teks untuk mengetik. Akhirnya, dokumentasi online jauh lebih murah untuk didistribusikan daripada dokumentasi kertas.

Jenis Dokumentasi

Ada tiga jenis dokumentasi pengguna yang berbeda secara mendasar: dokumen referensi, manual prosedur, dan tutorial. Dokumen referensi (juga disebut sistem bantuan) dirancang untuk digunakan ketika pengguna perlu mempelajari cara melakukan fungsi tertentu (mis., memperbarui bidang, menambahkan catatan baru). Seringkali orang membaca informasi referensi ketika mereka mencoba dan gagal menjalankan fungsinya; menulis

dokumen referensi memerlukan perhatian khusus karena pengguna sering kali tidak sabar atau frustrasi ketika dia mulai membacanya.

Manual prosedur menjelaskan bagaimana melakukan tugas bisnis (misalnya, mencetak laporan bulanan, menerima pesanan pelanggan). Setiap item dalam manual prosedur biasanya memandu pengguna melalui tugas yang memerlukan beberapa fungsi atau langkah dalam sistem. Oleh karena itu, setiap entri biasanya lebih panjang daripada entri dalam dokumen referensi.

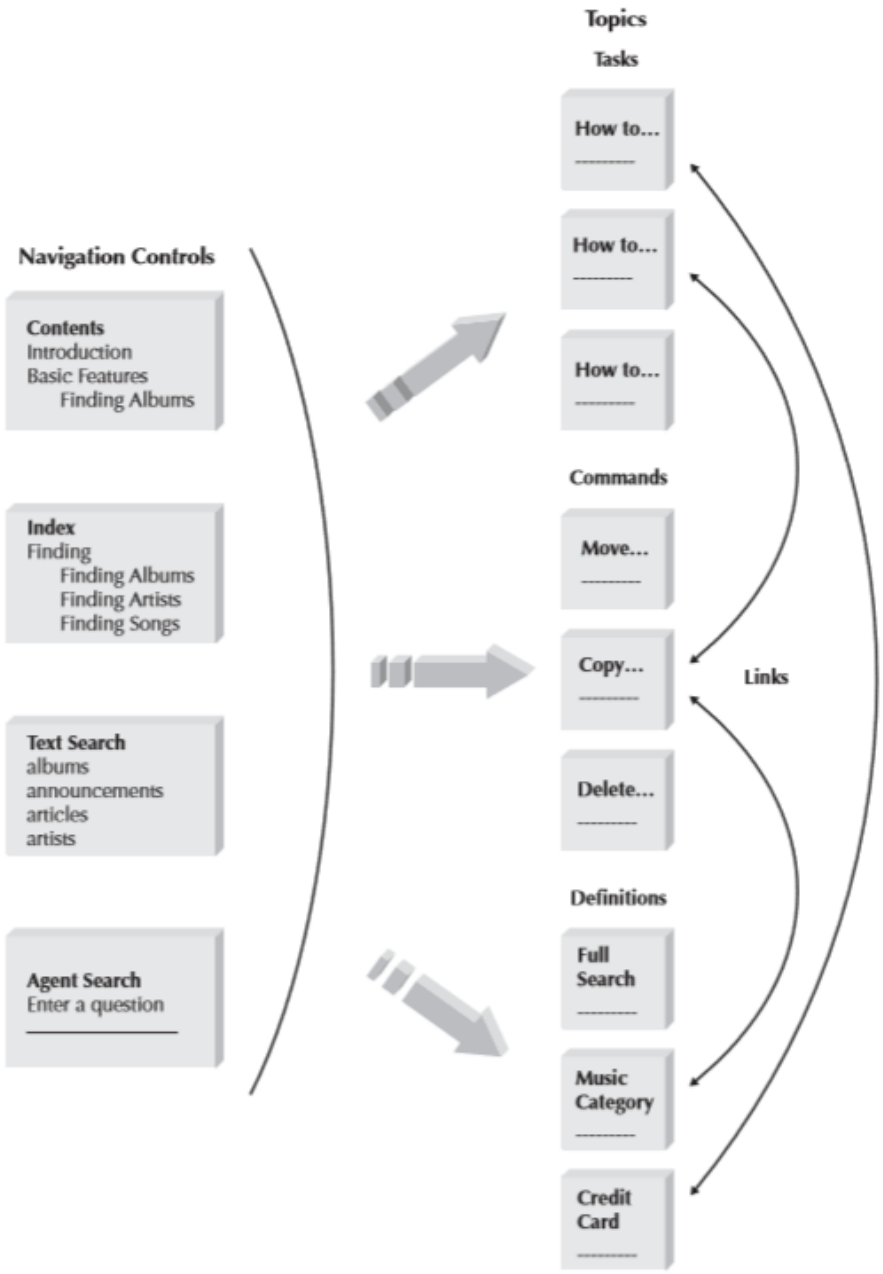
Tutorial—jelas—mengajarkan orang bagaimana menggunakan komponen utama dari suatu sistem (misalnya, pengenalan operasi dasar sistem). Setiap entri dalam tutorial biasanya lebih panjang daripada entri dalam manual prosedur, dan entri biasanya dirancang untuk dibaca secara berurutan (sedangkan entri dalam dokumen referensi dan manual prosedur dirancang untuk dibaca secara individual).

Terlepas dari jenis dokumentasi pengguna, keseluruhan proses pengembangannya mirip dengan proses pengembangan antarmuka (lihat Bab 10). Pengembang pertama-tama mendesain struktur umum untuk dokumentasi dan kemudian mengembangkan komponen individual di dalamnya.

Merancang Struktur Dokumentasi

Struktur umum yang digunakan di sebagian besar dokumentasi online, baik dokumen referensi, manual prosedur, atau tutorial, adalah untuk mengembangkan satu set kontrol navigasi dokumentasi yang mengarahkan pengguna ke topik dokumentasi. Topik dokumentasi adalah materi yang ingin dibaca pengguna, sedangkan kontrol navigasi adalah cara pengguna menemukan dan mengakses topik tertentu. Dengan demikian, menggunakan storyboard, diagram navigasi windows, dan diagram tata letak windows berguna (lihat Bab 10).

Merancang struktur dokumentasi dimulai dengan mengidentifikasi berbagai jenis topik dan kontrol navigasi yang perlu disertakan. Gambar 12-1 menunjukkan struktur yang umum digunakan untuk dokumen referensi online (yaitu, sistem bantuan). Topik dokumentasi umumnya berasal dari tiga sumber. Sumber topik pertama dan paling jelas adalah kumpulan perintah dan menu di antarmuka pengguna. Kumpulan topik ini sangat berguna jika pengguna ingin memahami bagaimana perintah atau menu tertentu digunakan.



Gambar 12-1 Mengatur Dokumen Referensi Online

Namun, pengguna sering tidak tahu perintah apa yang harus dicari atau di mana mereka berada dalam struktur menu sistem. Sebaliknya, pengguna memiliki tugas yang ingin mereka lakukan, dan alih-alih berpikir dalam hal perintah, mereka berpikir dalam hal tugas mereka. Oleh karena itu, topik yang kedua dan seringkali lebih berguna berfokus pada bagaimana melakukan tugas-tugas tertentu, biasanya dalam skenario penggunaan, WND, dan Use-Case nyata dari desain antarmuka pengguna (lihat Bab 10). Topik-topik ini memandu pengguna melalui serangkaian langkah (seringkali melibatkan beberapa penekanan tombol atau klik mouse) yang diperlukan untuk melakukan beberapa tugas.

Topik ketiga adalah definisi istilah-istilah penting. Istilah-istilah ini biasanya Use-Case dan kelas dalam sistem, tetapi kadang-kadang mereka juga menyertakan perintah.

Ada lima tipe umum kontrol navigasi untuk topik, tetapi tidak semua sistem menggunakan kelima tipe tersebut (lihat Gambar 12-1). Yang pertama adalah daftar isi yang mengatur informasi dalam bentuk logis, seolah-olah pengguna membaca dokumentasi

referensi dari awal sampai akhir. Indeks menyediakan akses ke topik berdasarkan kata utama penting, dengan cara yang sama seperti indeks di bagian belakang buku membantu kita menemukan topik. Pencarian teks menyediakan kemampuan untuk menelusuri topik baik untuk teks apa pun yang diketik pengguna atau kata-kata yang cocok dengan kumpulan kata yang ditentukan pengembang yang jauh lebih besar daripada kumpulan kata dalam indeks. Berbeda dengan indeks, pencarian teks biasanya tidak memberikan organisasi kata-kata (selain abjad). Beberapa sistem menyediakan kemampuan untuk menggunakan agen cerdas untuk membantu pencarian. Kontrol navigasi kelima dan terakhir untuk topik adalah hyperlink antara topik yang memungkinkan pengguna untuk mengklik dan berpindah di antara topik.

Prosedur manual dan tutorial serupa tetapi seringkali lebih sederhana dalam struktur. Topik untuk manual prosedur biasanya berasal dari skenario penggunaan, WND, dan Use-Case nyata yang dikembangkan selama desain antarmuka dan dari tugas dasar lainnya yang harus dilakukan pengguna. Topik untuk tutorial biasanya diatur di sekitar bagian utama dari sistem dan tingkat pengalaman pengguna. Kebanyakan tutorial dimulai dengan perintah dasar yang paling umum digunakan dan kemudian beralih ke perintah yang lebih kompleks dan jarang digunakan.

Menulis Topik Dokumentasi

Format umum untuk topik cukup mirip di seluruh sistem aplikasi dan sistem operasi. Topik biasanya dimulai dengan judul yang sangat jelas, diikuti oleh beberapa teks Pendahuluan yang mendefinisikan topik dan kemudian dengan petunjuk langkah demi langkah yang terperinci tentang cara melakukan apa yang sedang dijelaskan. Banyak topik menyertakan gambar layar untuk membantu pengguna menemukan item di layar; beberapa juga memiliki tutorial dan video yang tersedia secara online yang menunjukkan fungsi yang menarik bagi pengguna. Sebagian besar juga menyertakan kontrol navigasi untuk mengaktifkan perpindahan di antara topik, biasanya di bagian atas jendela, ditambah tautan ke topik lain. Beberapa juga menyertakan tautan ke topik terkait yang menyertakan opsi atau perintah dan tugas lain yang mungkin ingin dilakukan pengguna bersamaan dengan topik yang sedang dibaca.

Menulis konten topik bisa jadi menantang. Ini membutuhkan pemahaman yang baik tentang pengguna (atau lebih tepatnya kisaran pengguna) dan pengetahuan tentang keterampilan apa yang dimiliki pengguna saat ini dan dapat diharapkan untuk diimpor dari sistem dan alat lain yang mereka gunakan atau telah digunakan (termasuk sistem yang sistem baru menggantikan). Topik harus selalu ditulis dari sudut pandang pengguna dan menjelaskan apa yang ingin dicapai pengguna, bukan apa yang dapat dilakukan sistem. Gambar 12-2 memberikan beberapa panduan umum untuk meningkatkan kualitas teks dokumentasi.

Mengidentifikasi Istilah Navigasi

Saat kami menulis topik dokumentasi, kami juga mulai mengidentifikasi istilah yang akan digunakan untuk membantu pengguna menemukan topik. Daftar isi biasanya paling sederhana, karena dikembangkan dari struktur logis topik dokumentasi, baik topik referensi, topik prosedur, atau topik tutorial. Item untuk indeks dan mesin pencari memerlukan perawatan lebih karena dikembangkan dari bagian utama sistem dan fungsi bisnis pengguna. Setiap kali kita menulis suatu topik, kita juga harus mencantumkan istilah-istilah yang akan digunakan untuk mencari topik tersebut. Istilah untuk indeks dan mesin pencari dapat berasal dari empat sumber berbeda.

Pedoman	Sebelum menggunakan Pedoman	Setelah menggunakan Pedoman
Gunakan kalimat aktif: Kalimat aktif membuat teks lebih aktif dan mudah dibaca dengan meletakkan subjek di awal kalimat, kata kerja di tengah, dan objek di akhir.	Pencarian album dilakukan dengan menggunakan judul album, nama artis, atau judul lagu.	Anda dapat menemukan album dengan menggunakan judul album, nama artis, atau judul lagu.
Gunakan gaya e-prime: Gaya e-prime membuat tulisan lebih aktif dengan menghilangkan semua bentuk kata kerja to be.	Teks yang ingin Anda salin harus dipilih sebelum Anda mengklik tombol salin.	Pilih teks yang ingin Anda salin sebelum Anda mengklik tombol salin.
Gunakan istilah yang konsisten: Selalu gunakan istilah yang sama untuk merujuk ke item yang sama, jangan beralih pada sinonim (misalnya, mengubah, memodifikasi, memperbarui).	Pilih teks yang ingin Anda salin. Menekan tombol salin akan menyalin teks yang ditandai ke lokasi baru.	Pilih teks yang ingin Anda salin. Menekan tombol salin akan menyalin teks yang dipilih ke lokasi baru.
Gunakan bahasa yang sederhana: Selalu gunakan bahasa yang paling sederhana untuk menyampaikan makna secara akurat. Hal ini tidak berarti Anda harus "membodohi" teks tetapi Anda harus menghindari kompleksitasnya secara artifisial. Hindari memisahkan subjek dan kata kerja dan cobalah menggunakan kata-kata sesedikit mungkin. (Ketika Anda menemukan teks yang rumit, cobalah menghilangkan kata-katanya; Anda mungkin terkejut melihat betapa sedikit kata yang benar-benar diperlukan untuk menyampaikan suatu makna.)	*Georgia Statewide Academic and Medical System* (GSAMS) adalah jaringan pembelajaran jarak jauh yang kooperatif dan kolaboratif di negara bagian Georgia. Organisasi di Atlanta yang mengelola dan mengelola operasi teknis dan keseluruhan lebih dari 300 ruang kelas konferensi audio dan video interaktif di seluruh sistem Georgia adalah Department of Administrative Service (DOAS). (56 kata)	Department of Administrative Service (DOAS) di Atlanta mengelola Georgia Statewide Academic and Medical System (GSAMS), jaringan pembelajaran jarak jauh dengan lebih dari 300 ruang kelas telekonferensi di seluruh Georgia. (29 kata)
Gunakan bahasa yang ramah: Seringnya, dokumentasi dingin dan steril karena ditulis dengan cara yang sangat formal.	Disk kosong telah disediakan untuk Anda oleh bagian Operasi. Disarankan agar Anda memastikan data Anda tidak hilang dengan	Anda harus membuat salinan cadangan dari semua data yang penting bagi Anda. Jika Anda membutuhkan lebih banyak

Ingat, Anda menulis untuk seseorang, bukan untuk komputer.	membuat salinan cadangan semua data penting.	disket, hubungi bagian Operasi.
Gunakan struktur gramatikal paralel: Struktur gramatikal paralel menunjukkan kesamaan di antara item dalam daftar dan membantu pembaca memahami konten.	Membuka file Menyimpan dokumen Cara menghapus file	Membuka file Menyimpan file Menghapus file
Gunakan langkah-langkah dengan benar: Pemula sering kali menyela tindakan dan hasil dari suatu tindakan saat menjelaskan mengenai proses langkah demi langkah. Langkah selalu merupakan tindakan.	1. Tekan tombol pelanggan. 2. Akan muncul kotak dialog pelanggan. 3. Ketik ID pelanggan dan tekan tombol kirim dan catatan pelanggan akan muncul.	1. Tekan tombol pelanggan. 2. Ketik ID pelanggan di kotak dialog pelanggan ketika muncul. 3. Tekan tombol kirim untuk melihat catatan pelanggan untuk pelanggan ini.
Gunakan paragraf pendek: Pembaca dokumentasi biasanya dengan cepat memindai teks untuk menemukan informasi yang mereka butuhkan, sehingga teks di tengah paragraf panjang sering terlewatkan. Gunakan paragraf terpisah untuk membantu pembaca menemukan informasi lebih cepat.		
Sumber: Berdasarkan materi dari T. T. Barker, Writing Software Documentation (Boston: Allyn & Bacon, 1998).		

Gambar 12-2 Pedoman Penyusunan Topik Dokumentasi

Sumber pertama untuk istilah indeks adalah kumpulan perintah di antarmuka pengguna, seperti membuka file, memodifikasi pelanggan, dan mencetak pesanan terbuka. Semua perintah berisi dua bagian (aksi dan objek). Penting untuk mengembangkan indeks untuk kedua bagian karena pengguna dapat mencari informasi menggunakan salah satu bagian. Pengguna yang mencari informasi selengkapnya tentang menyimpan file, misalnya, mungkin mencari dengan menggunakan istilah simpan atau file istilah.

Sumber kedua adalah kumpulan konsep utama dalam sistem, yang sering menggunakan kasus dan kelas. Dalam kasus sistem Pengangkatan, misalnya, ini mungkin termasuk janji temu, gejala, atau pasien.

Sumber ketiga adalah serangkaian tugas bisnis yang dilakukan pengguna, seperti memesan unit pengganti atau membuat janji. Seringkali ini terkandung dalam kumpulan perintah, tetapi terkadang mereka memerlukan beberapa perintah dan menggunakan istilah yang tidak selalu muncul di sistem. Sumber yang baik untuk istilah-istilah ini adalah skenario penggunaan dan Use-Case nyata yang dikembangkan selama desain antarmuka (lihat Bab 10)

Sumber keempat, seringkali kontroversial, adalah kumpulan sinonim untuk tiga kumpulan item sebelumnya. Pengguna terkadang tidak memikirkan istilah yang didefinisikan dengan baik yang digunakan oleh sistem. Mereka mungkin mencoba mencari informasi tentang cara menghentikan atau berhenti daripada keluar, atau menghapus daripada menghapus. Memasukkan sinonim dalam indeks meningkatkan kompleksitas dan ukuran sistem dokumentasi tetapi dapat sangat meningkatkan kegunaan sistem bagi pengguna.

12.5 TES PERANCANGAN

Dalam sistem berorientasi objek, godaannya adalah meminimalkan pengujian. Bagaimanapun, melalui penggunaan pola, kerangka kerja, pustaka kelas, dan komponen, banyak dari sistem telah diuji sebelumnya. Oleh karena itu, kita tidak harus menguji terlalu banyak. Benar? Salah! Pengujian lebih penting untuk sistem berorientasi objek daripada sistem yang dikembangkan di masa lalu. Berdasarkan enkapsulasi (dan penyembunyian informasi), polimorfisme (dan pengikatan dinamis), pewarisan, penggunaan kembali, dan produk berorientasi objek yang sebenarnya, pengujian menyeluruh jauh lebih sulit dan kritis. Mengingat kompleksitas proses pengembangan yang digunakan dan sifat global pengembangan sistem informasi, pengujian menjadi lebih penting. Dengan demikian, pengujian berorientasi objek harus dilakukan secara sistematis, dan hasilnya harus didokumentasikan agar tim proyek mengetahui apa yang telah dan belum diuji. Oleh karena itu, pengujian sistem berorientasi objek sangat kompleks. Akibatnya, cakupan topik yang lengkap berada di luar cakupan buku ini.

Tujuan pengujian bukan untuk menunjukkan bahwa sistem bebas dari kesalahan. Tidaklah mungkin untuk membuktikan bahwa suatu sistem bebas dari kesalahan. Tujuan pengujian adalah untuk mengungkap perbedaan antara apa yang sebenarnya dilakukan sistem dan apa yang seharusnya dilakukan sistem. Dengan kata lain, tujuan pengujian adalah untuk mencoba dan merusak sistem. Ini mirip dengan pengujian teori. Anda tidak dapat membuktikan teori. Jika tes gagal menemukan masalah dengan teori, kepercayaan Anda pada teori meningkat. Namun, jika suatu tes berhasil menemukan suatu masalah, maka teori tersebut telah dipalsukan. Pengujian perangkat lunak serupa karena hanya dapat menunjukkan adanya kesalahan. Jadi, inti dari pengujian adalah untuk mengungkap kesalahan sebanyak mungkin. Sama sekali tidak hemat biaya untuk mencoba menghilangkan setiap kesalahan dari perangkat lunak. Kecuali dalam contoh sederhana, pada kenyataannya, itu tidak mungkin. Ada terlalu banyak kombinasi untuk diperiksa.

Ada empat tahap umum pengujian: pengujian unit, pengujian integrasi, pengujian sistem, dan pengujian penerimaan. Meskipun setiap sistem aplikasi berbeda, kebanyakan kesalahan ditemukan selama integrasi dan pengujian sistem. Selain tahapan pengujian yang berbeda, pengujian harus memenuhi persyaratan fungsional dan nonfungsional. Namun, sebelum masuk ke jenis pengujian tertentu, kami menjelaskan efek yang dimiliki karakteristik berorientasi objek pada pengujian dan aktivitas perencanaan dan manajemen yang diperlukan yang harus dilakukan agar program pengujian berhasil.

Pengujian dan Orientasi Objek

Sebagian besar teknik pengujian telah dikembangkan untuk mendukung pengembangan yang tidak berorientasi objek. Oleh karena itu, sebagian besar pendekatan pengujian harus disesuaikan dengan sistem berorientasi objek. Karakteristik sistem berorientasi objek yang paling mempengaruhi pengujian adalah enkapsulasi (dan penyembunyian informasi); polimorfisme (dan pengikatan dinamis); warisan; dan penggunaan pola, perpustakaan kelas, kerangka kerja, dan komponen. Juga, banyaknya produk yang keluar dari proses pengembangan berorientasi objek yang khas telah meningkatkan pentingnya pengujian dalam pengembangan sistem berorientasi objek.

Enkapsulasi dan Penyembunyian Informasi. Enkapsulasi dan penyembunyian informasi memungkinkan proses dan data digabungkan untuk membuat entitas holistik (yaitu, objek). Mereka mendukung menyembunyikan segala sesuatu di balik antarmuka yang terlihat. Meskipun hal ini memungkinkan sistem untuk dimodifikasi dan dipelihara dengan cara yang efektif dan efisien, hal ini membuat pengujian sistem menjadi bermasalah. Apa yang perlu Anda uji untuk membangun kepercayaan pada kemampuan sistem untuk memenuhi kebutuhan pengguna? Anda perlu menguji proses bisnis yang diwakili dalam Use-Case. Namun, proses bisnis didistribusikan melalui sekumpulan kelas yang berkolaborasi dan terkandung dalam metode kelas tersebut. Satu-satunya cara untuk mengetahui pengaruh proses bisnis terhadap suatu sistem adalah dengan melihat perubahan keadaan yang terjadi dalam sistem. Tetapi dalam sistem berorientasi objek, instance dari kelas menyembunyikan data di balik batas kelas. Lalu bagaimana mungkin untuk melihat dampak dari proses bisnis?

Masalah kedua yang diangkat oleh enkapsulasi dan penyembunyian informasi adalah definisi "unit" untuk pengujian unit. Apa unit yang akan diuji? Apakah paket, kelas, atau metode? Dalam pendekatan tradisional, jawabannya adalah proses yang terkandung dalam suatu fungsi. Namun, proses dalam sistem berorientasi objek didistribusikan melalui satu set kelas. Oleh karena itu, menguji metode individual tidak masuk akal. Jawabannya adalah kelas. Ini secara dramatis mengubah cara pengujian unit dilakukan.

Isu ketiga yang diangkat adalah dampak pada pengujian integrasi. Dalam hal ini, objek dapat digabungkan untuk membentuk objek agregat; misalnya, mobil memiliki banyak bagian, atau mereka dapat dikelompokkan bersama untuk membentuk kolaborasi. Selanjutnya, mereka dapat digunakan di perpustakaan kelas, kerangka kerja, dan komponen. Berdasarkan semua cara yang berbeda ini kelas dapat dikelompokkan bersama, bagaimana seseorang secara efektif melakukan pengujian integrasi?

Polimorfisme dan Pengikatan Dinamis. Polimorfisme dan pengikatan dinamis secara dramatis memengaruhi pengujian unit dan integrasi. Karena proses bisnis individu diimplementasikan melalui serangkaian metode yang didistribusikan di atas sekumpulan objek, seperti yang ditunjukkan sebelumnya, pengujian unit tidak masuk akal pada tingkat metode. Namun, dengan polimorfisme dan pengikatan dinamis, metode yang sama (sebagian kecil dari keseluruhan proses bisnis) dapat diimplementasikan di banyak objek yang berbeda. Oleh karena itu, menguji implementasi metode individual tidak masuk akal. Sekali lagi, unit yang masuk akal untuk diuji adalah kelasnya. Kecuali untuk kasus-kasus sepele, pengikatan dinamis tidak memungkinkan untuk mengetahui implementasi mana yang akan dieksekusi sampai sistem melakukannya. Oleh karena itu, pengujian integrasi menjadi sangat menantang.

Warisan Ketika mempertimbangkan masalah yang diangkat tentang pewarisan (lihat Bab 8), seharusnya tidak mengejutkan bahwa pewarisan mempengaruhi pengujian sistem berorientasi objek. Melalui penggunaan pewarisan, bug dapat disebarkan secara instan dari superclass ke semua subclass langsung dan tidak langsung. Namun, tes yang berlaku untuk superclass juga berlaku untuk semua subclass-nya. Seperti biasa, warisan adalah pedang bermata dua. Akhirnya, meskipun kami telah menyatakan ini berkali-kali sebelumnya, pewarisan seharusnya hanya mendukung jenis semantik generalisasi dan spesialisasi. Ingat, ketika menggunakan pewarisan, prinsip substitusi sangat penting (lihat Bab 5). Semua masalah ini memengaruhi pengujian unit dan integrasi.

Penggunaan kembali. Di permukaan, penggunaan kembali harus mengurangi jumlah pengujian yang diperlukan. Namun, setiap kali sebuah kelas digunakan dalam konteks yang berbeda, kelas tersebut harus diuji kembali. Oleh karena itu, setiap kali perpustakaan kelas, kerangka kerja, atau komponen digunakan, pengujian unit dan pengujian integrasi adalah penting. Dalam hal komponen, unit yang akan diuji adalah komponen itu sendiri. Ingat bahwa

komponen memiliki API (antarmuka program aplikasi) yang terdefinisi dengan baik yang menyembunyikan detail implementasinya.

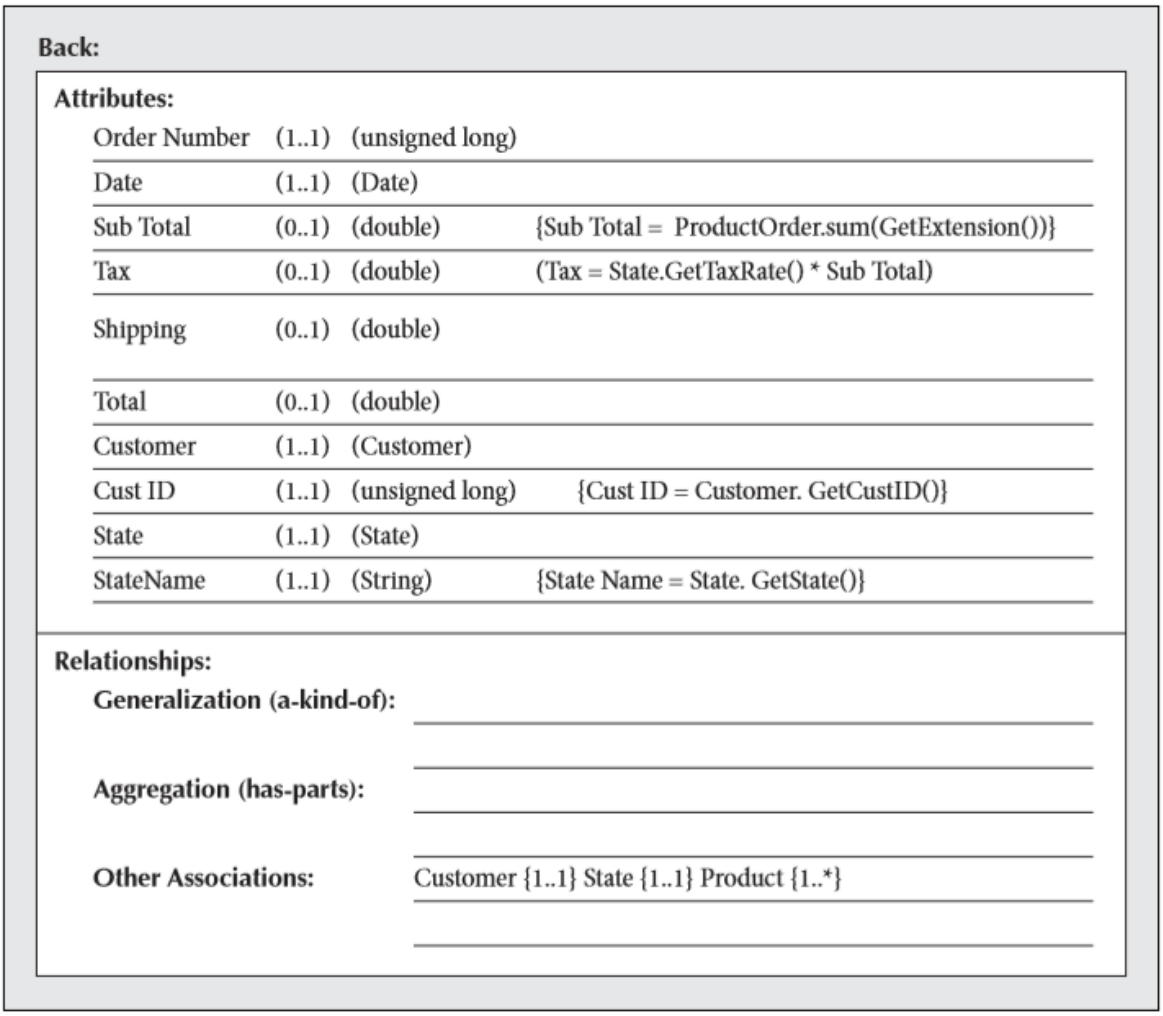
Proses dan Produk Pengembangan Berorientasi Objek. Di hampir semua buku teks, termasuk yang ini, pengujian dibahas menjelang akhir pengembangan sistem. Ini tampaknya menyiratkan bahwa pengujian adalah sesuatu yang terjadi hanya setelah pemrograman berakhir. Namun, setiap produk yang keluar dari proses pengembangan berorientasi objek harus diuji. Misalnya, jauh lebih mudah untuk memastikan bahwa persyaratan ditangkap dan dimodelkan dengan benar melalui pengujian Use-Case, dan jauh lebih murah untuk menangkap jenis kesalahan ini kembali dalam analisis daripada dalam implementasi. Jelas, ini juga berlaku untuk menguji kolaborasi. Pada saat kita mengimplementasikan kolaborasi sebagai satu set layer dan partisi, kita bisa menghabiskan banyak waktu—dan waktu adalah uang—untuk mengimplementasikan hal yang salah. Jadi menguji kolaborasi dengan memainkan peran kartu CRC dalam analisis sebenarnya menghemat banyak waktu dan uang tim.

Front:

Class Name: Order	**ID:** 2	**Type:** Concrete, Domain
Description: An Individual who needs to receive or has received medical attention		**Associated Use Cases:** 3

Responsibilities	**Collaborators**
Calculate subtotal	
Calculate tax	
Calculate shipping	
Calculate total	

(a)



(b)

Gambar 12-3 Memesan Kartu CRC (lihat Gambar 8-19)

Pengujian adalah sesuatu yang harus dilakukan selama pengembangan sistem, bukan hanya di akhir. Namun, jenis pengujian yang dapat dilakukan pada representasi yang tidak dapat dieksekusi, seperti Use-Case dan kartu CRC, berbeda dari pengujian pada kode yang ditulis dalam bahasa pemrograman berorientasi objek. Pendekatan utama untuk menguji representasi yang tidak dapat dieksekusi adalah beberapa bentuk inspeksi atau penelusuran representasi. Dalam bab-bab sebelumnya, kami berfokus pada verifikasi dan validasi berbagai analisis dan representasi desain. Kami juga memastikan bahwa representasi yang berbeda konsisten dan seimbang. Dengan demikian, kami telah berurusan dengan pengujian representasi nonexecutable selama proses pengembangan (lihat Bab 4-11).

Perencanaan Tes

Pengujian dimulai dengan pengembangan rencana pengujian, yang mendefinisikan serangkaian pengujian yang akan dilakukan. Karena pengujian berlangsung selama pengembangan sistem berorientasi objek, rencana pengujian harus dikembangkan pada awal pengembangan sistem dan terus diperbarui seiring perkembangan sistem. Misalnya, representasi kelas berkembang dari kartu CRC sederhana menjadi satu set kelas yang diimplementasikan dalam bahasa pemrograman.

Pada Gambar 12-3 kita melihat representasi kartu CRC dari kelas Order yang berisi invarian. Masing-masing invarian ini harus diuji dan ditegakkan agar kelas Pesanan dianggap

memiliki kualitas yang memadai. Satu tes invarian sederhana adalah mencoba menetapkan nilai ke atribut ID Cust yang tidak terkait dengan objek Pelanggan yang terkandung dalam atribut Pelanggan. Tes invarian lainnya adalah mencoba dan menetapkan lebih dari satu tanggal ke atribut Date. Akhirnya, tes invarian yang lebih rumit adalah mencoba menetapkan nilai integer ke atribut Pengiriman. Yang ini lebih sulit karena sebagian besar bahasa pemrograman mengizinkan bilangan bulat untuk "dilemparkan" menjadi ganda. Jika nilai yang terkandung dalam atribut Pengiriman benar-benar seharusnya ganda, maka casting nilai integer ke ganda akan menjadi kesalahan. Tes ini harus dilakukan dengan menggunakan pendekatan walkthrough ketika kelas ditentukan, seperti yang kita lakukan di Bab 4, 5, 6, dan 7, dan pendekatan yang lebih ketat setelah kelas sepenuhnya diterapkan. Ini adalah contoh pengujian unit sebuah kelas, yang akan dijelaskan nanti dalam bab ini. Untuk memastikan kualitas kelas, itu harus diuji setiap kali representasinya diubah.

Rencana pengujian harus membahas semua produk yang dibuat selama pengembangan sistem. Misalnya, tes harus dibuat yang dapat digunakan untuk menguji kelengkapan kartu CRC. Setiap tes individu memiliki tujuan khusus dan menggambarkan serangkaian kasus uji yang sangat spesifik untuk diperiksa. Dalam kasus tes berbasis invarian, deskripsi invarian diberikan, dan nilai asli atribut, peristiwa yang akan menyebabkan nilai atribut berubah, hasil aktual yang diamati, hasil yang diharapkan, dan apakah lulus atau tidak. gagal ditampilkan. Spesifikasi pengujian dibuat untuk setiap jenis kendala yang harus dipenuhi oleh kelas. Juga, jenis spesifikasi yang serupa dilakukan untuk pengujian integrasi, sistem, dan penerimaan.

Tidak semua kelas kemungkinan akan selesai pada saat yang sama, sehingga programmer biasanya menulis stub untuk kelas yang belum selesai untuk memungkinkan kelas di sekitar mereka untuk diuji. Sebuah rintisan adalah pengganti untuk kelas yang biasanya menampilkan pesan pengujian sederhana di layar atau mengembalikan beberapa nilai hardcode saat dipilih. Misalnya, pertimbangkan sistem aplikasi yang menyediakan fungsi membuat, mengubah, menghapus, menemukan, dan mencetak untuk beberapa objek seperti CD, pasien, atau karyawan. Tergantung pada desain akhir, fungsi yang berbeda ini dapat berakhir di objek yang berbeda pada lapisan yang berbeda. Oleh karena itu, untuk menguji fungsionalitas yang terkait dengan kelas pada lapisan domain masalah, sebuah rintisan akan ditulis untuk setiap kelas pada lapisan lain yang berinteraksi dengan kelas domain masalah. Rintisan ini akan menjadi antarmuka minimal yang diperlukan untuk dapat menguji kelas domain masalah. Misalnya, mereka akan memiliki metode yang dapat menerima pesan yang dikirim oleh objek lapisan domain masalah dan metode yang dapat mengirim pesan ke objek lapisan domain masalah. Biasanya, metode akan menampilkan pesan di layar yang memberi tahu penguji bahwa metode tersebut berhasil dicapai (mis., Hapus item dari metode Database tercapai). Dengan cara ini, kelas domain masalah dapat lulus pengujian kelas sebelum kelas pada lapisan lain diselesaikan.

Akhirnya, seperti yang Anda duga, perencanaan pengujian harus dilakukan selama proses pengembangan. Jauh lebih mudah untuk merancang tes ketika Anda membuat analisis dan representasi desain yang berbeda daripada menunggu dan mendesainnya selama konstruksi sistem.

Tes Unit

Tes unit fokus pada satu unit—kelas. Ada dua pendekatan untuk pengujian unit: pengujian kotak hitam dan pengujian kotak putih (lihat Gambar 12-4). Pengujian kotak hitam adalah yang paling umum digunakan karena setiap kelas mewakili objek yang dienkapsulasi. Pengujian kotak hitam didorong oleh kartu CRC, mesin status perilaku, dan kontrak yang terkait dengan kelas, bukan oleh interpretasi pemrogram. Dalam hal ini, rencana tes dikembangkan langsung dari spesifikasi kelas: setiap item dalam spesifikasi menjadi tes, dan

beberapa kasus tes dikembangkan untuk itu. Pengujian kotak putih didasarkan pada spesifikasi metode yang terkait dengan setiap kelas. Namun, pengujian kotak putih memiliki dampak terbatas dalam pengembangan berorientasi objek. Hal ini disebabkan ukuran yang agak kecil dari metode individu dalam kelas. Sebagian besar pendekatan untuk menguji kelas menggunakan pengujian kotak hitam untuk memastikan kebenarannya.

Tahap	**Tipe Pengujian**	**Sumber Rencana Pengujian**	**Kapan Menggunakan**	**Catatan**
Pengujian Unit	Pengujian Black-Box Memperlakukan kelas sebagai kotak hitam	Kartu CRC Diagram Kelas Kontrak	Untuk pengujian unit normal	• Penguji fokus pada apakah kelas memenuhi persyaratan yang dinyatakan dalam spesifikasi.
	Pengujian White-Box Melihat ke dalam kelas untuk menguji elemen utamanya	Spesifikasi Metode	Ketika kompleksitas tinggi	• Dengan melihat ke dalam kelas untuk meninjau kode itu sendiri, penguji mungkin menemukan kesalahan atau asumsi yang tidak langsung terlihat oleh seseorang yang memperlakukan kelas sebagai kotak hitam.
Pengujian integrasi	Pengujian Antarmuka Pengguna Penguji menguji setiap fungsi antarmuka	Desain Antarmuka	Untuk pengujian integrasi normal	• Pengujian dilakukan dengan menelusuri setiap item menu di antarmuka baik secara top-down atau bottom-up.
	Pengujian Penggunaan Kasus Penguji menguji setiap kasus penggunaan	Kasus Penggunaan	Ketika antarmuka pengguna penting	• Pengujian dilakukan dengan menelusuri setiap kasus penggunaan untuk memastikan bahwa antarmuka tsb bekerja dengan benar. • Biasanya dikombinasikan dengan pengujian antarmuka pengguna karena tidak menguji semua antarmuka.

	Pengujian Interaksi Menguji setiap proses dengan selangkah demi selangkah	Diagram Kelas Diagram Berurutan Diagram Komunikasi	Ketika sistem melakukan data processing	• Seluruh sistem dimulai sebagai satu set rintisan. Setiap kelas ditambahkan secara bergiliran dan hasil kelas dibandingkan dengan hasil yang benar dari data uji; ketika sebuah kelas lolos, kelas berikutnya ditambahkan dan tes dijalankan kembali. Hal ini dilakukan untuk setiap paket. Setelah setiap paket berhasil melewati semua tes, maka proses mengulangi untuk mengintegrasikan paket.
	Pengujian Antarmuka Sistem Menguji pertukaran data dengan sistem lain	Diagram Penggunaan Kasus	Ketika sistem melakukan pertukaran data	• Karena transfer data antar sistem sering kali otomatis dan tidak dipantau secara langsung oleh pengguna, sangat penting untuk merancang pengujian untuk memastikan bahwa pertukaran dilakukan dengan benar.
Pengujian Sistem	Pengujian Persyaratan Pengujian mengenai apakah persyaratan bisnis asli terpenuhi	Desain Sistem, Pengujian Unit, dan Pengujian Integrasi	Untuk pengujian sistem normal	• Memastikan bahwa perubahan yang dibuat berdasarkan hasil dari pengujian integrasi tidak menyebabkan kesalahan baru. • Penguji dapat berpura-pura menjadi pengguna yang tidak mendapat informasi dan melakukan tindakan yang tidak tepat untuk memastikan bahwa sistem kebal terhadap tindakan yang tidak

				valid (misalnya, menambahkan catatan kosong).
	Pengujian Kegunaan Menguji seberapa nyaman sistem digunakan	Desain Antarmuka dan Kasus Penggunaan	Ketika antarmuka pengguna penting	• Sering dilakukan oleh analis yang berpengalaman mengenai cara berpikir pengguna dan desain antarmuka yang baik. • Kadang-kadang menggunakan prosedur pengujian kegunaan formal yang dibahas dalam bab sebelumnya.
	Pengujian Dokumentasi Menguji keakuratan dokumentasi	Sistem Bantuan, Prosedur, Tutorial	Untuk pengujian sistem normal	• Analis memeriksa titik atau memeriksa setiap item pada setiap halaman di semua dokumentasi untuk memastikan bahwa item dan contoh dokumentasi berfungsi dengan baik.
	Pengujian Kinerja Memeriksa kemampuan untuk bekerja di bawah beban tinggi	Proposal Sistem Desain Infrastruktur	Ketika sistem penting	• Volume transaksi yang tinggi dihasilkan dan diberikan ke sistem. • Sering dilakukan dengan menggunakan perangkat lunak pengujian bertujuan khusus.
	Pengujian Keamanan Menguji pemulihan bencana dan akses tidak sah	Desain Infrastruktur	Ketika sistem penting	• Pengujian keamanan adalah tugas kompleks yang biasanya dilakukan oleh analis infrastruktur yang ditugaskan untuk proyek tersebut. • Dalam kasus ekstrim, sebuah perusahaan profesional dapat dipekerjakan.
Pengujian Penerimaan	Pengujian Alfa Dilakukan oleh pengguna untuk memastikan	Pengujian Sistem	Untuk pengujian penerimaan normal	• Sering mengulangi tes sebelumnya tetapi dilakukan oleh pengguna sendiri

	bahwa mereka menerima sistem			untuk memastikan bahwa antarmuka telah menerima sistem.
	Pengujian Beta	Persyaratan Sistem	Ketika sistem penting	• Pengguna memonitor sistem untuk melihat kesalahan atau perbaikan yang diperlukan.

Gambar 12-4 Jenis Tes

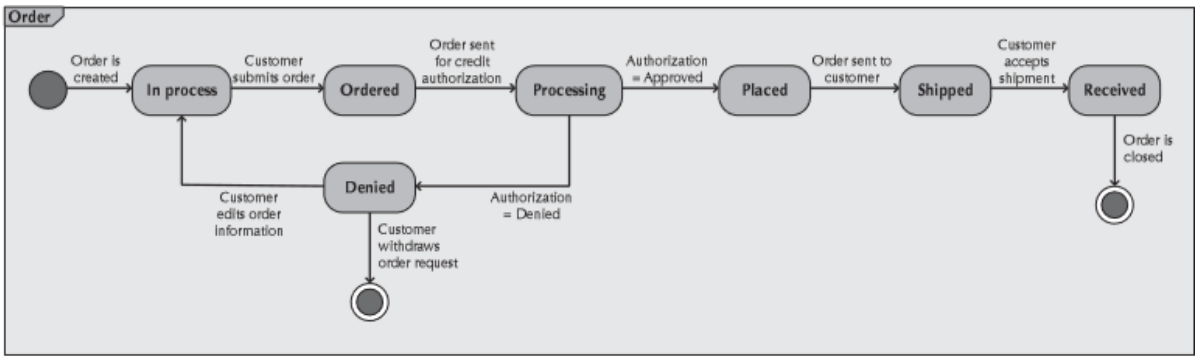
Tes kelas harus didasarkan pada invarian pada kartu CRC, mesin status perilaku yang terkait dengan setiap kelas, dan kondisi pra dan pasca yang terdapat pada kontrak setiap metode. Dengan asumsi semua kendala telah ditangkap pada kartu dan kontrak CRC, kasus uji individu dapat dikembangkan dengan cukup mudah. Misalnya, kartu CRC untuk kelas pesanan memberikan invarian bahwa jumlah pesanan harus antara 10 dan 100 kasus. Penguji akan mengembangkan serangkaian kasus uji untuk memastikan bahwa kuantitas divalidasi sebelum sistem menerimanya. Tidak mungkin menguji setiap kemungkinan kombinasi input dan situasi; ada terlalu banyak kemungkinan kombinasi. Dalam contoh ini, pengujian memerlukan minimal tiga kasus uji: satu dengan nilai valid (mis., 15), satu dengan nilai valid rendah (mis., 7), dan satu dengan nilai valid tinggi (mis., 110). Sebagian besar pengujian juga akan menyertakan kasus uji dengan nilai nonnumerik untuk memastikan tipe data diperiksa (misalnya, ABCD). Tes yang benar-benar baik akan mencakup kasus uji dengan data yang tidak masuk akal tetapi berpotensi valid (mis. 21.4).

Menggunakan mesin status perilaku adalah cara yang berguna untuk mengidentifikasi tes untuk suatu kelas. Setiap kelas yang memiliki mesin status perilaku yang terkait dengannya memiliki siklus hidup yang berpotensi kompleks. Dimungkinkan untuk membuat serangkaian tes untuk menjamin bahwa setiap negara bagian dapat dicapai. Misalnya, Gambar 12-5 menggambarkan mesin status perilaku untuk kelas Order yang baru saja dibahas. Dalam hal ini, ada banyak transisi antara status yang berbeda dari sebuah instance dari kelas Order. Pengujian harus dibuat untuk menjamin bahwa satu-satunya transisi yang diizinkan dari turunan kelas Order adalah transisi yang ditentukan secara spesifik. Dalam kasus ini, objek Pesanan tidak mungkin beralih dari status Dalam proses ke status Ditempatkan tanpa melintasi status Dipesan dan Memproses melalui Pelanggan mengirimkan pesanan, Pesanan dikirim untuk otorisasi kredit, dan Otorisasi = transisi yang Disetujui. Pengujian berbasis status ini dapat dilakukan selama pengembangan kelas melalui penelusuran dan permainan peran di awal evolusi kelas dan pengujian yang lebih ketat setelah diimplementasikan dalam bahasa pemrograman.

Tes juga dapat dikembangkan untuk setiap kontrak yang terkait dengan kelas. Dalam hal kontrak, diperlukan serangkaian pengujian untuk setiap pra-kondisi dan pasca-kondisi. Misalnya, kontrak metode add Order dari kelas Pelanggan yang ditunjukkan pada Gambar 12-6 memiliki kondisi pra dan pasca yang pada dasarnya mengharuskan agar pesanan baru tidak ada dengan instance kelas Pelanggan sebelum metode dijalankan dan pesanan baru dikaitkan dengan objek Pelanggan setelah metode dijalankan. Tes harus dibuat untuk menegakkan batasan ini. Jika kelas tersebut merupakan subkelas dari kelas lain, maka semua pengujian yang terkait dengan superkelas tersebut harus dijalankan kembali. Interaksi antara kendala, invarian, dan kondisi sebelum dan sesudah di subkelas dan superkelas juga harus ditangani.

Akhirnya, karena desain berorientasi objek yang baik, untuk sepenuhnya menguji kelas, metode pengujian khusus mungkin harus ditambahkan ke kelas yang diuji. Misalnya, bagaimana invarian dapat diuji? Satu-satunya cara untuk benar-benar mengujinya adalah

dengan memiliki metode yang terlihat di luar kelas yang dapat digunakan untuk memanipulasi nilai atribut kelas. Namun, menambahkan jenis metode ini ke kelas melakukan dua hal. Pertama, mereka menambah persyaratan pengujian karena mereka sendiri harus diuji. Kedua, jika mereka tidak dihapus dari versi sistem yang digunakan, sistem akan menjadi kurang efisien, dan keuntungan dari penyembunyian informasi secara efektif akan hilang. Seperti yang sudah terlihat, kelas pengujian itu kompleks. Oleh karena itu, perhatian besar harus diberikan ketika merancang tes untuk kelas.



Gambar 12-5 Order Behavioral State Machine (lihat Gambar 6-18)

Nama Metode : addOrder	Nama Kelas : Customer	ID : 36
Client (Costumer) :		
Use-Case Terkait: addCustomerOrder		
Deskripsi Tanggung Jawab: Terapkan perilaku yang diperlukan untuk menambahkan pesanan baru ke pelanggan yang sudah ada dengan menyimpan pesanan dalam urutan yang diurutkan berdasarkan nomor pesanan.		
Argumen yang Diterima: anOrder:Order		
Jenis Nilai yang Dikembalikan: void		
Pre-Conditions: Not Order.includes(anOrder)		

Pre-Conditions:

Orders = Orders@pre.including(anOrder)

Gambar 12-6 Kontrak AddOrder (lihat Gambar 8-25)

Tes Integrasi

Tes integrasi menilai apakah sekumpulan kelas yang harus bekerja sama melakukannya tanpa kesalahan. Mereka memastikan bahwa antarmuka dan hubungan antara bagian yang berbeda dari sistem bekerja dengan baik. Pada titik ini, kelas telah lulus tes unit masing-masing, jadi fokusnya sekarang adalah pada aliran kontrol di antara kelas dan pada data yang dipertukarkan di antara mereka. Pengujian integrasi mengikuti prosedur umum yang sama dengan pengujian unit: Penguji mengembangkan rencana pengujian yang memiliki serangkaian pengujian, yang, pada gilirannya, memiliki pengujian. Pengujian integrasi sering dilakukan oleh sekelompok programmer dan/atau analis sistem.

Dari perspektif sistem berorientasi objek, pengujian integrasi bisa jadi sulit. Satu kelas dapat berada dalam banyak agregasi yang berbeda, karena cara objek dapat digabungkan untuk membentuk objek baru, pustaka kelas, kerangka kerja, komponen, dan paket. Di mana tempat terbaik untuk memulai integrasi? Biasanya, jawabannya adalah memulai dengan kumpulan kelas, sebuah kolaborasi, yang digunakan untuk mendukung Use-Case dengan prioritas tertinggi (lihat Bab 4). Juga, pengikatan dinamis membuatnya penting untuk merancang tes integrasi dengan hati-hati untuk memastikan bahwa kombinasi metode diuji.

Ada empat pendekatan untuk pengujian integrasi: pengujian antarmuka pengguna,16 pengujian Use-Case, pengujian interaksi, dan pengujian antarmuka sistem (lihat Gambar 12-4). Sebagian besar proyek menggunakan keempat pendekatan tersebut. Namun, seperti pengujian unit, pengujian integrasi harus direncanakan dengan cermat. Dalam kasus pengujian Use-Case, hanya aspek kelas dan invarian kelas yang terkait dengan Use-Case khusus yang disertakan dalam pengujian kelas tergantung konteks Use-Case ini. Faktanya, biasanya pengujian Use-Case dilakukan satu skenario pada satu waktu. Dalam banyak hal, pengujian Use-Case dapat dilihat sebagai latihan bermain peran yang lebih ketat (lihat Bab 5). Seperti pengujian unit, pengujian integrasi harus dilakukan sepanjang evolusi sistem. Pada tahap awal pengembangan sistem, Anda harus bekerja dengan kartu CRC dan memainkannya. Nanti, Anda akan menyelesaikan kontrak dan spesifikasi metode. Secara bertahap, Anda akan mengimplementasikan kelas domain masalah, kelas antarmuka pengguna, dan kelas lapisan manajemen data dalam bahasa pemrograman. Seperti dalam pengujian unit, setiap kali representasi baru (diagram, teks, program) dibuat, pengujian integrasi baru perlu dilakukan. Oleh karena itu, seiring dengan berkembangnya sistem untuk lebih mendukung use case, kita dapat menguji dengan lebih teliti apakah use case didukung sepenuhnya atau tidak.

Salah satu masalah utama dengan pengujian integrasi dan sistem berorientasi objek adalah kesulitan yang disebabkan oleh interaksi pewarisan dan pengikatan dinamis. Masalah khusus ini telah dikenal sebagai masalah yo-yo. Masalah yo-yo terjadi ketika analis atau desainer harus melambung ke atas dan ke bawah melalui grafik pewarisan untuk memahami aliran kontrol melalui metode yang dijalankan. Dalam kebanyakan kasus, ini disebabkan oleh grafik pewarisan yang agak dalam; yaitu, subclass memiliki banyak superclass di atasnya dalam grafik pewarisan. Masalah yo-yo menjadi lebih dari mimpi buruk dalam pengujian sistem berorientasi objek ketika ada konflik pewarisan dan ketika pewarisan berganda

digunakan (lihat Bab 8). Tentang satu-satunya pendekatan realistis untuk pengujian melalui masalah yo-yo adalah melalui debugger interaktif yang biasanya merupakan bagian dari lingkungan pengembangan sistem, seperti Eclipse, Netbeans, atau Visual Studio.

Tes Sistem

Untuk memastikan bahwa semua kelas bekerja sama tanpa kesalahan, analis sistem biasanya melakukan pengujian sistem. Pengujian sistem mirip dengan pengujian integrasi tetapi cakupannya jauh lebih luas. Sedangkan pengujian integrasi berfokus pada apakah kelas bekerja sama tanpa kesalahan, pengujian sistem memeriksa seberapa baik sistem memenuhi persyaratan fungsional dan nonfungsional, misalnya, kegunaan, dokumentasi, kinerja, dan keamanan (lihat Gambar 12-4).

Tujuan pengujian persyaratan fungsional adalah untuk memastikan bahwa persyaratan fungsional yang ditemukan memang terpenuhi. Seperti pengujian integrasi, ini terutama didorong oleh Use-Case sistem dan skenarionya. Namun, dalam banyak kasus, pengujian integrasi memerlukan modifikasi pada sistem. Jadi, fokus dari requirement testing adalah untuk memastikan bahwa modifikasi yang dilakukan tidak menimbulkan error tambahan.

Pengujian kegunaan pada dasarnya adalah kombinasi dari antarmuka pengguna dan pengujian Use-Case yang terjadi selama pengujian integrasi. Di mana pengujian antarmuka pengguna dan Use-Case berfokus pada apakah antarmuka pengguna berfungsi dan apakah Use-Case didukung, masing-masing, pengujian kegunaan berfokus pada seberapa baik antarmuka pengguna mendukung Use-Case. Artinya, seberapa efisien dan efektif antarmuka pengguna. Dalam banyak kasus, ini dapat mencakup pengujian kegunaan formal (lihat Bab 10).

Mengingat bahwa dokumentasi pada dasarnya adalah sebuah sistem itu sendiri, pengujian dokumentasi harus melibatkan pengujian unit dan integrasi. Dalam hal ini, unit adalah entri dokumentasi, dan antarmuka pengguna adalah kertas atau layar bantuan. Dari perspektif pengujian integrasi, fokusnya adalah pada apakah dokumentasi berfungsi atau tidak. Dan, seperti pengujian sistem perangkat lunak, fokus pengujian sistem dokumentasi adalah seberapa baik dokumentasi bekerja. Alasan bahwa dokumentasi biasanya tidak diuji secara paralel dengan sistem, yaitu ketika kelas, Use-Case, dan antarmuka pengguna diuji, adalah untuk meminimalkan jumlah pengujian dokumentasi yang diperlukan. Meskipun dokumentasi harus dikembangkan secara paralel dengan perangkat lunak, sampai perangkat lunak diuji, tidak jelas apa yang harus diuji dalam dokumentasi. Saat perangkat lunak “lulus” dalam pengujiannya, dokumentasi yang menyertai perangkat lunak tersebut kemudian dapat diselesaikan dan diuji.

Pengujian kinerja berfokus pada upaya untuk merusak sistem sehubungan dengan jumlah pekerjaan yang dapat ditangani sistem. Jenis tes ini biasanya terbagi dalam dua kategori: tes stres dan tes volume. Tujuan dari stress test, juga dikenal sebagai load test, adalah untuk memastikan bahwa sistem dapat menangani sejumlah permintaan simultan. Misalnya, jika sistem seharusnya mampu menangani 10.000 permintaan simultan, tes stres akan mencoba mendorong sistem untuk menangani lebih dari itu. Jika kinerja pengujian tidak mencukupi, berbagai optimasi perangkat lunak dan database (lihat Bab 8 dan 9) dapat diselidiki. Dalam kasus lain, perangkat keras tambahan mungkin diperlukan. Tujuan dari tes volume adalah untuk mendorong implementasi sehingga dapat rusak ketika ada sejumlah besar data yang diperlukan untuk menjawab permintaan pengguna. Sekali lagi, jika ditemukan bahwa sistem gagal dalam pengujian jenis ini, maka pengoptimalan database dan perangkat lunak serta perangkat keras tambahan mungkin diperlukan. Misalnya, terkadang lebih efisien untuk membuat sekumpulan tabel sementara dengan "memilih" data dari tabel yang sebenarnya sebelum "menggabungkan" tabel bersama-sama. Dengan melakukan "memilih"

terlebih dahulu, "bergabung" bekerja pada lebih sedikit data. Dalam hal ini, ini dapat mempercepat dan mengurangi jumlah penyimpanan sementara yang diperlukan untuk menangani permintaan. Dalam kasus lain, denormalisasi data dan penyimpanan data di beberapa lokasi dapat dilakukan. Anda biasanya tidak ingin pengguna membuat permintaan laporan dan harus menunggu "terlalu lama" untuk memproses laporan. Dalam beberapa kasus, melepaskan beberapa fungsionalitas untuk meningkatkan kinerja dapat menjadi sangat penting bagi keberhasilan sistem. Jadi, hasil pengujian kinerja dapat membuat atau menghancurkan suatu sistem.

Jelas, di dunia jaringan saat ini, pengujian keamanan sangat penting. Pengujian keamanan melibatkan tiga bidang utama: otentikasi, otorisasi, dan pengendalian virus. Pengujian otentikasi berkaitan dengan memastikan bahwa pengguna yang masuk adalah siapa yang dia klaim. Biasanya, ini telah diatasi dengan ID pengguna dan kata sandi dan melalui penggunaan teknik enkripsi (lihat Bab 11). Saat ini, selain pendekatan ini, berbagai pengenal biometrik telah digunakan, misalnya, pemindaian retina dan sidik jari. Pengujian otorisasi berkaitan dengan memastikan bahwa pengguna yang masuk benar-benar memiliki wewenang untuk menggunakan sistem yang sedang diakses. Otorisasi telah dikendalikan melalui penggunaan peran, daftar kontrol akses, dan daftar kemampuan. Peran keamanan sama dengan peran aktor dalam model Use-Case. Tergantung pada peran yang dimainkan oleh pengguna, kemampuan yang berbeda tersedia untuk pengguna dalam bentuk daftar kemampuan. Namun, dalam hal ini, peran dapat ditentukan hingga ke tingkat pengguna individu dan tidak terbatas pada sekelompok pengguna. Juga, daftar kontrol akses dapat dikaitkan dengan setiap Use-Case dan dengan setiap kelas. Dalam hal ini, daftar kontrol akses menentukan peran mana yang memiliki akses ke sumber daya (Use-Case atau kelas). Mengingat bahwa banyak pembobolan sistem adalah fungsi dari virus, pengendalian virus juga perlu ditegakkan. Setiap kali file diterima atau dikirim oleh pengguna, file tersebut harus dipindai untuk kemungkinan virus. Ini termasuk lampiran email, unduhan Web, dan penyisipan flash drive di komputer desktop serta semua bentuk mesin "klien" yang dapat dilampirkan ke sistem. Jelas, persyaratan keamanan akan berdampak pada kinerja sistem. Oleh karena itu, trade-off antara dua set persyaratan ini mungkin diperlukan.

Tes Penerimaan

Pengujian penerimaan dilakukan terutama oleh pengguna dengan dukungan dari tim proyek. Tujuannya adalah untuk mengkonfirmasi bahwa sistem sudah lengkap, memenuhi kebutuhan bisnis yang mendorong sistem untuk dikembangkan, dan dapat diterima oleh pengguna. Pengujian penerimaan dilakukan dalam dua tahap: pengujian alfa, di mana pengguna menguji sistem menggunakan data yang dibuat-buat, dan pengujian beta, di mana pengguna mulai menggunakan sistem dengan data nyata tetapi dimonitor secara hati-hati untuk kesalahan (lihat Gambar 12-4).

Pertanyaan

1. Mengapa pengujian itu penting?
2. Bagaimana budaya nasional atau organisasi yang berbeda dapat mempengaruhi manajemen proyek pengembangan sistem informasi?
3. Apa peran utama analis sistem selama tahap pemrograman?
4. Dalam The Mythical Man-Month, Frederick Brooks berpendapat bahwa menambahkan lebih banyak programmer ke proyek yang terlambat membuatnya nanti. Mengapa?
5. Ketika pengembangan lepas pantai, bagaimana perbedaan dalam dimensi konteks budaya Hall dapat mempengaruhi kontribusi anggota tim terhadap keberhasilan pengembangan sistem informasi? Bagaimana dengan waktu atau kecepatan pesan Hall?

6. Apa lima dimensi perbedaan budaya Hofstede? Bagaimana perbedaan di dalamnya dapat mempengaruhi efektivitas tim pengembangan sistem informasi?
7. Apa bahasa atau bahasa yang umum digunakan saat ini dalam pengembangan sistem informasi?
8. Bandingkan dan kontraskan dokumentasi pengguna dan dokumentasi sistem.
9. Mengapa dokumentasi online menjadi lebih penting?
10. Apa kelemahan utama dokumentasi online?
11. Bandingkan dan kontraskan dokumen referensi, manual prosedur, dan tutorial.
12. Apa lima jenis navigasi dokumentasi?
13. 13. Apa sumber topik dokumentasi yang umum digunakan? Mana yang paling penting? Mengapa?
14. Apa sumber kontrol navigasi dokumentasi yang umum digunakan? Mana yang paling penting? Mengapa?
15. Apa tujuan dari pengujian?
16. Jelaskan bagaimana orientasi objek mempengaruhi pengujian.
17. Bandingkan dan bedakan istilah uji, rencana uji, dan kasus uji.
18. Apa itu rintisan dan mengapa digunakan dalam pengujian?
19. Apa tujuan utama dari pengujian unit?
20. Bagaimana kasus uji dikembangkan untuk pengujian unit?
21. Bandingkan dan kontraskan pengujian kotak hitam dan pengujian kotak putih.
22. Apa saja jenis-jenis tes kelas?
23. Apa tujuan utama dari pengujian integrasi?
24. Bagaimana kasus uji dikembangkan untuk uji integrasi?
25. Jelaskan masalah yo-yo. Mengapa itu membuat pengujian integrasi menjadi sulit?
26. Apa tujuan utama dari pengujian sistem?
27. Bagaimana kasus uji dikembangkan untuk pengujian sistem?
28. Apa tujuan utama dari pengujian penerimaan?
29. Bagaimana kasus uji dikembangkan untuk uji penerimaan?
30. Bandingkan dan kontraskan pengujian alfa dan pengujian beta.

Latihan

A. Pandangan yang berbeda tentang plagiarisme dan pembelajaran kolaboratif digambarkan sebagai contoh perbedaan di antara budaya yang berbeda saat ini. Menggunakan Web, mengidentifikasi perbedaan lain yang dapat mempengaruhi keberhasilan tim pengembangan sistem informasi.
B. Selain Hall dan Hofstede, baik David Victor dan Fons Trompenaars telah mengidentifikasi seperangkat dimensi budaya yang dapat berguna dalam pengembangan sistem informasi. Menggunakan Web, mengidentifikasi dimensi mereka.
C. Jika sistem pendaftaran di universitas Anda tidak memiliki sistem bantuan online yang baik, kembangkan satu untuk satu layar antarmuka pengguna.
D. Periksa dan siapkan laporan tentang sistem bantuan online untuk program kalkulator di Windows (atau yang serupa di Mac atau Unix). (Anda mungkin akan terkejut dengan jumlah bantuan untuk program yang begitu sederhana.)
E. Bandingkan dan kontraskan bantuan online di dua situs web berbeda yang memungkinkan Anda melakukan beberapa fungsi (misalnya, membuat reservasi perjalanan, memesan buku).
F. Buat spesifikasi tes invarian untuk kelas yang Anda pilih untuk masalah A Real Estate Inc. dalam latihan A di Bab 8.
G. Buat rencana uji Use-Case, termasuk rencana kelas khusus dan pengujian invarian, untuk Use-Case dari latihan A Real Estate Inc. di bab sebelumnya.

H. Buat spesifikasi tes invarian untuk kelas yang Anda pilih untuk masalah A Video Store dalam latihan B di Bab 8.
I. Buat rencana pengujian Use-Case, termasuk paket kelas khusus dan pengujian invarian, untuk Use-Case dari latihan A Video Store di bab sebelumnya.
J. Buat spesifikasi tes invarian untuk kelas yang Anda pilih untuk masalah gym dalam latihan C di Bab 8.
K. Buat rencana uji Use-Case, termasuk rencana kelas khusus dan tes invarian untuk Use-Case dari latihan klub kesehatan di bab sebelumnya.
L. Buat spesifikasi tes invarian untuk kelas yang Anda pilih untuk Picnics R Us dalam latihan D di Bab 8.
M. Buat rencana uji Use-Case, termasuk rencana kelas khusus dan pengujian invarian, untuk Use-Case dari latihan Piknik R Us di bab sebelumnya.
N. Buat spesifikasi pengujian invarian untuk kelas yang Anda pilih untuk Klub Bulanan (OTMC) dalam latihan E di Bab 8.
O. Buat rencana pengujian Use-Case, termasuk rencana kelas spesifik dan pengujian invarian, untuk penggunaan kasus dari latihan Of-the-Month Club (OTMC) di bab-bab sebelumnya.

Minicase

1. Pete adalah manajer proyek pada proyek pengembangan sistem baru. Proyek ini adalah pengalaman pertama Pete sebagai manajer proyek, dan dia telah memimpin timnya dengan sukses ke tahap pemrograman proyek. Proyek ini tidak selalu berjalan mulus, dan Pete telah membuat beberapa kesalahan, tetapi secara umum dia senang dengan kemajuan timnya dan kualitas sistem yang dikembangkan. Sekarang pemrograman telah dimulai, Pete berharap untuk sedikit istirahat dalam kesibukan hari kerjanya.

   Sebelum memulai pemrograman, Pete menyadari bahwa perkiraan waktu yang dibuat sebelumnya dalam proyek terlalu optimis. Namun, dia berkomitmen kuat untuk memenuhi tenggat waktu proyek karena keinginannya agar proyek pertamanya sebagai manajer proyek sukses. Untuk mengantisipasi masalah tekanan waktu ini, Pete mengatur dengan departemen Sumber Daya Manusia untuk mendatangkan dua lulusan perguruan tinggi baru dan dua magang perguruan tinggi untuk menambah staf pemrograman. Pete ingin mencari beberapa staf dengan lebih banyak pengalaman, tetapi anggarannya terlalu ketat, dan dia berkomitmen untuk menjaga agar anggaran proyek tetap terkendali.

   Pete membuat tugas pemrogramannya, dan mengerjakan program itu dimulai sekitar dua minggu yang lalu. Sekarang, Pete mulai mendengar beberapa keributan dari pemimpin tim pemrograman yang mungkin menandakan masalah. Tampaknya para pemrogram telah melaporkan beberapa contoh di mana mereka menulis program, hanya untuk tidak dapat menemukannya ketika mereka pergi untuk mengujinya. Juga, beberapa programmer telah membuka program yang mereka tulis, hanya untuk menemukan bahwa seseorang telah mengubah bagian dari program mereka tanpa sepengetahuan mereka.

   a. Apakah fase pemrograman suatu proyek merupakan waktu bagi manajer proyek untuk bersantai? Mengapa atau mengapa tidak?
   b. Masalah apa yang dapat Anda identifikasi dalam situasi ini?
   c. Apa saran Anda untuk manajer proyek? Seberapa besar kemungkinan Pete akan mencapai tujuan yang diinginkannya tepat waktu dan sesuai anggaran jika tidak ada yang dilakukan?

2. Analis sistem sedang mengembangkan rencana pengujian untuk antarmuka pengguna untuk sistem Kendaraan Perjalanan Liburan. Saat wiraniaga memasukkan faktur penjualan ke dalam sistem, mereka akan dapat memasukkan kode opsi ke dalam kotak teks atau memilih kode opsi dari daftar drop-down. Kotak kombo digunakan untuk menerapkan ini, karena dirasakan bahwa tenaga penjualan akan dengan cepat menjadi akrab dengan kode opsi yang paling umum dan lebih suka memasukkannya secara langsung untuk mempercepat proses entri.

   Sekarang saatnya untuk mengembangkan tes untuk memvalidasi bidang kode opsi selama entri data. Jika pelanggan tidak meminta opsi yang dipasang dealer untuk kendaraan tersebut, penjual harus memasukkan "none"; kolom tidak boleh kosong. Kode opsi yang valid adalah kode alfabet empat karakter dan harus dicocokkan dengan daftar kode yang valid.

   Siapkan rencana pengujian untuk pengujian bidang kode opsi selama entri data.

BAB 13

INSTALASI DAN OPERASI

Bab ini membahas aktivitas yang diperlukan untuk menginstal sistem informasi dan berhasil mengubah organisasi untuk menggunakannya. Selain itu juga membahas kegiatan pasca implementasi, seperti dukungan sistem, pemeliharaan sistem, dan penilaian proyek. Menginstal sistem dan membuatnya tersedia untuk digunakan dari perspektif teknis relatif mudah. Namun, pelatihan dan masalah organisasi seputar penginstalan lebih kompleks dan menantang karena berfokus pada orang, bukan komputer.

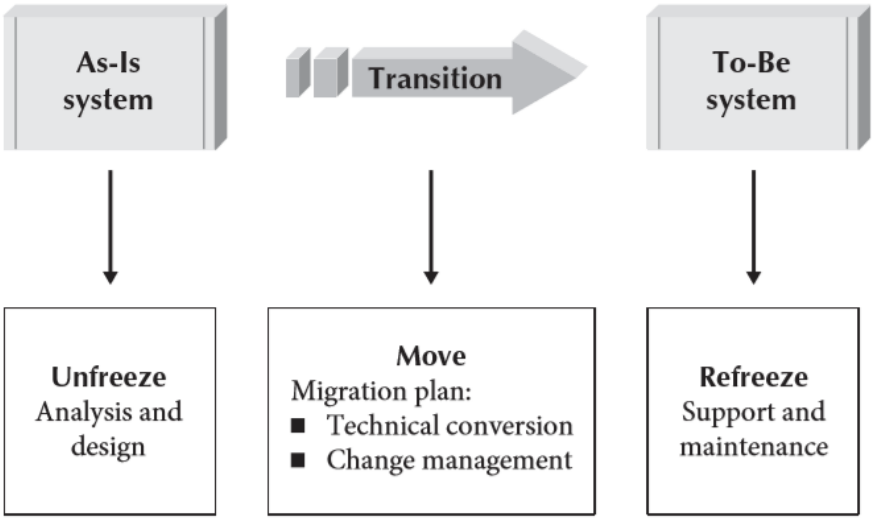
13.1 Tujuan

- Pahami proses instalasi sistem.
- Pahami berbagai jenis strategi konversi dan kapan menggunakannya.
- Memahami beberapa teknik untuk mengelola perubahan.
- Biasakan dengan proses pasca-instalasi.

13.2 Pendahuluan

Meskipun ditulis hampir 500 tahun yang lalu, komentar Machiavelli masih berlaku sampai sekarang. Mengelola perubahan ke sistem baru — apakah itu terkomputerisasi atau tidak — adalah salah satu tugas tersulit di organisasi mana pun. Karena tantangan yang ada, sebagian besar organisasi mulai mengembangkan rencana konversi dan manajemen perubahan mereka sementara pemrogram masih mengembangkan perangkat lunak. Membiarkan konversi dan mengubah perencanaan manajemen hingga menit terakhir adalah resep untuk kegagalan.

Dalam banyak hal, menggunakan sistem komputer atau serangkaian proses kerja sama seperti mengemudi di jalan tanah. Seiring waktu, dengan penggunaan berulang, jalan mulai membentuk bekas roda di bagian jalan yang paling umum digunakan. Meskipun bekas roda ini menunjukkan ke mana harus mengemudi, mereka membuat perubahan menjadi sulit. Ketika orang menggunakan sistem komputer atau serangkaian proses kerja, sistem atau proses kerja tersebut mulai menjadi kebiasaan atau norma; orang mempelajarinya dan menjadi nyaman dengannya. Sistem atau proses kerja ini kemudian mulai membatasi aktivitas orang dan mempersulit mereka untuk berubah karena mereka mulai melihat pekerjaan mereka dalam hal proses ini daripada tujuan bisnis akhir melayani pelanggan.



Gambar 13-1 Menerapkan Perubahan

Salah satu model paling awal untuk mengelola perubahan organisasi dikembangkan oleh Kurt Lewin. Lewin berpendapat bahwa perubahan adalah proses tiga langkah: unfreeze, move, refreeze (Gambar 13-1). Pertama, tim proyek harus mencairkan kebiasaan dan norma yang ada (sistem apa adanya) sehingga perubahan dapat dilakukan. Sebagian besar pengembangan sistem hingga saat ini telah meletakkan dasar untuk pencairan. Pengguna menyadari sistem baru yang sedang dikembangkan, beberapa telah berpartisipasi dalam analisis sistem saat ini (dan juga menyadari masalahnya), dan beberapa telah membantu merancang sistem baru (dan memiliki beberapa pengertian tentang manfaat potensial dari sistem baru tersebut). sistem baru). Kegiatan ini telah membantu mencairkan kebiasaan dan norma saat ini.

Langkah kedua adalah membantu organisasi pindah ke sistem baru melalui rencana migrasi. Rencana migrasi memiliki dua elemen utama. Salah satunya adalah teknis, yang mencakup bagaimana sistem baru akan diinstal dan bagaimana data dalam sistem saat in akan dipindahkan ke sistem yang akan datang; ini dibahas di bagian konversi bab ini. Komponen kedua adalah organisasi, yang mencakup membantu pengguna memahami perubahan dan memotivasi mereka untuk mengadopsinya; ini dibahas di bagian manajemen perubahan bab ini.

Langkah ketiga adalah membekukan kembali sistem baru sebagai cara biasa melakukan proses kerja—memastikan bahwa sistem baru berhasil menjadi cara standar untuk menjalankan fungsi bisnis yang didukungnya. Proses refreezing ini adalah tujuan utama dari kegiatan pasca implementasi yang dibahas di bagian akhir bab ini. Dengan memberikan dukungan berkelanjutan untuk sistem baru dan segera mulai mengidentifikasi perbaikan untuk versi sistem berikutnya, organisasi membantu memperkuat sistem baru sebagai cara kebiasaan baru dalam melakukan bisnis. Kegiatan pasca implementasi meliputi dukungan sistem, yang berarti menyediakan help desk dan dukungan telepon bagi pengguna yang bermasalah; pemeliharaan sistem, yang berarti memperbaiki bug dan meningkatkan sistem setelah diinstal; dan penilaian proyek, mengevaluasi proyek untuk mengidentifikasi apa yang berjalan dengan baik dan apa yang dapat ditingkatkan untuk proyek pengembangan sistem berikutnya.

Manajemen perubahan adalah yang paling menantang dari tiga komponen karena berfokus pada orang, bukan teknologi, dan karena merupakan salah satu aspek proyek yang paling tidak dapat dikendalikan oleh tim proyek. Manajemen perubahan berarti memenangkan

hati dan pikiran pengguna potensial dan meyakinkan mereka bahwa sistem baru benar-benar memberikan nilai.

Pemeliharaan adalah aspek yang paling mahal dari proses instalasi, karena biaya pemeliharaan sistem biasanya jauh melebihi biaya pengembangan awal. Bukan hal yang aneh bagi organisasi untuk menghabiskan 60 hingga 80 persen dari total anggaran pengembangan SI mereka untuk pemeliharaan. Meskipun ini mungkin terdengar mengejutkan pada awalnya, pikirkan tentang perangkat lunak yang Anda gunakan. Berapa banyak paket perangkat lunak yang Anda gunakan yang merupakan versi pertama? Sebagian besar paket perangkat lunak komersial menjadi sangat berguna dan digunakan secara luas hanya dalam versi kedua atau ketiganya. Pemeliharaan dan peningkatan berkelanjutan dari perangkat lunak sedang berlangsung, apakah itu paket yang tersedia secara komersial atau perangkat lunak yang dikembangkan sendiri. Apakah Anda akan membeli perangkat lunak jika Anda tahu bahwa tidak ada versi baru yang akan diproduksi? Tentu saja, perangkat lunak komersial agak berbeda dari perangkat lunak internal yang digunakan hanya oleh satu perusahaan, tetapi masalah mendasar tetap ada.

Penilaian proyek mungkin merupakan bagian pengembangan sistem yang paling jarang dilakukan tetapi mungkin merupakan salah satu yang memiliki nilai paling jangka panjang bagi departemen SI. Penilaian proyek memungkinkan anggota tim proyek untuk mundur dan mempertimbangkan apa yang mereka lakukan dengan benar dan apa yang bisa mereka lakukan dengan lebih baik. Ini adalah komponen penting dalam pertumbuhan dan perkembangan individu setiap anggota tim, karena mendorong anggota tim untuk belajar dari keberhasilan dan kegagalan mereka. Ini juga memungkinkan ide-ide baru atau pendekatan baru untuk pengembangan sistem dikenali, diperiksa, dan dibagikan dengan tim proyek lain untuk meningkatkan kinerja mereka.

13.3 ISU BUDAYA DAN ADOPSI TEKNOLOGI INFORMASI

Isu-isu budaya adalah salah satu hal yang biasanya diidentifikasi sebagai setidaknya sebagian untuk disalahkan ketika ada kegagalan dalam suatu organisasi. Isu-isu budaya telah dipelajari di tingkat organisasi dan nasional. Dalam bab-bab sebelumnya, kita membahas efek yang dapat ditimbulkan oleh isu-isu budaya dalam merancang interaksi manusia-komputer dan lapisan arsitektur fisik (lihat Bab 10 dan 11) dan manajemen pemrogram (Bab 12). Dimensi budaya yang diidentifikasi oleh Hall dan Hofstede termasuk kecepatan pesan, konteks, waktu, jarak kekuasaan, penghindaran ketidakpastian, individualisme versus kolektivisme, maskulinitas versus feminitas, dan orientasi jangka panjang versus jangka pendek. Dalam bab ini, kami menjelaskan bagaimana dimensi ini dapat mempengaruhi keberhasilan penyebaran sistem informasi yang mendukung rantai pasokan informasi global.

Dimensi pertama Hall, kecepatan pesan, memiliki implikasi untuk pengembangan dokumentasi (lihat Bab 12) dan pendekatan pelatihan (lihat nanti di bab ini). Dalam budaya yang menghargai konten "dalam", sehingga anggota budaya dapat meluangkan waktu untuk memahami sistem baru secara menyeluruh, hanya menyediakan sistem bantuan online tidak akan cukup untuk memastikan keberhasilan penerapan sistem informasi baru. . Namun, dalam budaya yang lebih menyukai pesan “cepat”, sistem bantuan online sudah cukup.

Dimensi kedua Hall, konteks, juga mempengaruhi adopsi dan penyebaran sistem baru. Dalam budaya konteks tinggi, diharapkan sistem informasi baru akan ditempatkan ke seluruh konteks sistem perusahaan-lebar. Anggota masyarakat jenis ini berharap dapat memahami dengan tepat di mana sistem cocok dengan gambaran keseluruhan perusahaan. Sekali lagi, seperti dimensi kecepatan pesan, ini mempengaruhi pendekatan pelatihan yang digunakan dan dokumentasi yang dikembangkan.

Dimensi ketiga Hall, waktu, juga dapat mempengaruhi adopsi dan penyebaran sistem baru. Dalam budaya waktu polikronik, pelatihan mungkin perlu disebarkan dalam jangka waktu yang lebih lama, jika dibandingkan dengan budaya waktu monokronis. Dalam budaya waktu monokromatik, interupsi akan dianggap tidak sopan. Akibatnya, pelatihan dapat dicapai dalam satu set kecil sesi intens. Namun, dengan budaya waktu polikronik, karena interupsi mungkin sering terjadi, fleksibilitas maksimum dalam menyiapkan sesi pelatihan mungkin diperlukan.

Dimensi pertama Hofstede, jarak kekuasaan, membahas bagaimana masalah kekuasaan ditangani dalam budaya. Misalnya, jika seorang atasan dalam suatu organisasi memiliki keyakinan yang salah tentang suatu isu penting, dapatkah seorang bawahan menunjukkan kesalahan ini? Dalam beberapa budaya, jawabannya adalah tidak. Akibatnya, dimensi ini bisa memiliki konsekuensi besar bagi keberhasilan penyebaran sistem informasi. Misalnya, dalam budaya dengan jarak kekuasaan yang tinggi, penerapan sistem informasi baru bergantung pada kesan pemegang kepentingan yang paling penting (lihat Bab 2). Oleh karena itu, banyak perhatian harus diberikan untuk memastikan bahwa pemegang kepentingan ini senang dengan sistem tersebut. Jika tidak, itu mungkin tidak akan pernah digunakan.

Dimensi kedua Hofstede, penghindaran ketidakpastian, didasarkan pada sejauh mana budaya bergantung pada aturan arah, seberapa baik individu dalam budaya menangani stres, dan pentingnya stabilitas pekerjaan. Misalnya, dalam budaya penghindaran ketidakpastian tinggi, penggunaan manual prosedur terperinci (lihat Bab 12) dan pelatihan yang baik (lihat nanti dalam bab ini) dapat mengurangi ketidakpastian dalam mengadopsi sistem baru.

Dimensi ketiga Hofstede, individualisme versus kolektivisme, didasarkan pada tingkat penekanan budaya pada individu atau kolektif. Hubungan antara individu dan kelompok penting bagi keberhasilan suatu sistem informasi. Bergantung pada orientasi budaya, keberhasilan sistem informasi yang dialihkan ke produksi dapat bergantung pada apakah fokus sistem informasi akan menguntungkan individu atau kelompok.

Dimensi keempat Hofstede, maskulinitas versus feminitas, membahas seberapa baik karakteristik maskulin dan feminin dihargai oleh budaya. Beberapa perbedaan yang dapat mempengaruhi penerapan sistem informasi termasuk masalah motivasi karyawan. Dalam budaya maskulin, motivasi akan didasarkan pada kemajuan, penghasilan, dan pelatihan, sedangkan dalam budaya feminin, motivasi akan mencakup suasana bersahabat, kondisi fisik, dan kerja sama. Bergantung pada bagaimana budaya memandang dimensi ini, motivasi yang berbeda mungkin perlu digunakan untuk meningkatkan kemungkinan sistem informasi berhasil digunakan.

Dimensi kelima, orientasi jangka panjang versus jangka pendek, berkaitan dengan bagaimana budaya memandang masa lalu dan masa depan. Di Asia Timur, pemikiran jangka panjang sangat dihormati, sedangkan di Amerika Utara dan Eropa, keuntungan jangka pendek dan harga saham saat ini tampaknya menjadi satu-satunya hal yang penting. Berdasarkan dimensi ini, semua kepentingan politik yang diangkat sebelumnya dalam teks ini menjadi sangat penting. Misalnya, jika budaya lokal memandang kesuksesan hanya dalam jangka pendek, maka sistem informasi baru apa pun yang digunakan untuk mendukung satu departemen dalam suatu organisasi dapat memberi departemen itu keunggulan kompetitif atas departemen lain dalam jangka pendek. Jika hanya ukuran jangka pendek yang digunakan untuk menilai keberhasilan suatu departemen, maka departemen lain akan berkepentingan untuk melawan keberhasilan penyebaran sistem informasi. Namun, jika perspektif jangka panjang adalah norma, maka departemen lain dapat diyakinkan untuk mendukung sistem informasi baru karena mereka dapat memiliki sistem informasi pendukung baru di masa depan.

Jelas, ketika meninjau dimensi ini, kita dapat melihat mereka berinteraksi satu sama lain. Yang paling penting untuk diingat dari perspektif TI adalah kita harus berhati-hati untuk tidak melihat komunitas pengguna lokal melalui mata kita; dalam ekonomi global, kita harus mempertimbangkan masalah budaya lokal agar sistem informasi dapat digunakan dengan sukses.

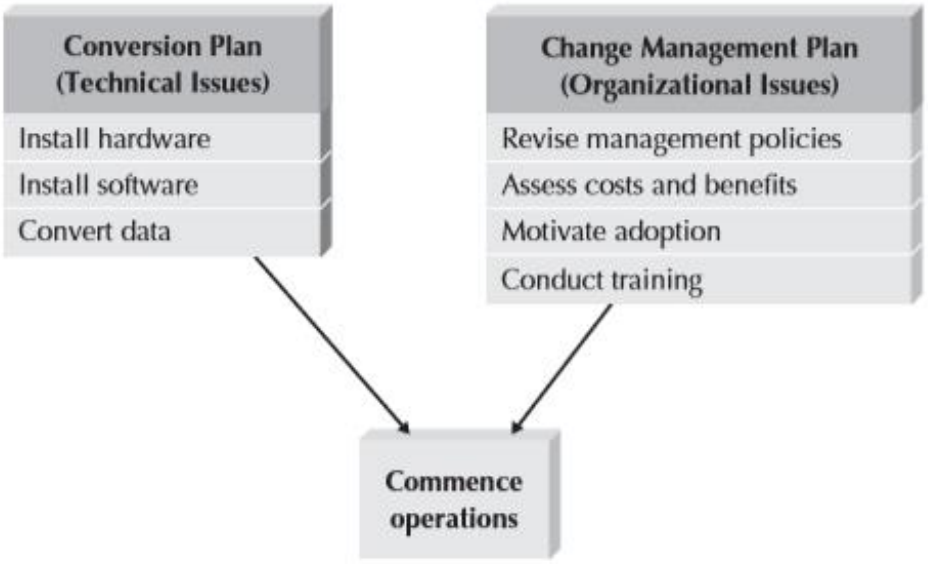
13.4 KONVERSI

Konversi adalah proses teknis dimana sistem baru menggantikan sistem lama. Pengguna dipindahkan dari menggunakan proses bisnis dan program komputer apa adanya ke proses dan program bisnis yang akan datang. Rencana migrasi menentukan kegiatan apa yang akan dilakukan kapan dan oleh siapa dan mencakup aspek teknis (seperti menginstal perangkat keras dan perangkat lunak dan mengubah data dari sistem saat in ke sistem yang akan datang) dan aspek organisasi (seperti pelatihan dan motivasi. pengguna untuk merangkul sistem baru). Konversi mengacu pada aspek teknis dari rencana migrasi.

Ada tiga langkah utama untuk rencana konversi sebelum memulai operasi: Menginstal perangkat keras, menginstal perangkat lunak, dan mengonversi data (Gambar 13-2). Meskipun mungkin untuk melakukan beberapa langkah ini secara paralel, biasanya mereka harus dilakukan secara berurutan di satu lokasi.

Langkah pertama dalam rencana konversi adalah membeli dan memasang perangkat keras yang diperlukan. Dalam banyak kasus, tidak diperlukan perangkat keras baru, tetapi terkadang proyek memerlukan perangkat keras baru seperti server, komputer klien, printer, dan peralatan jaringan. Sangat penting untuk bekerja sama dengan vendor yang memasok perangkat keras dan perangkat lunak yang dibutuhkan untuk memastikan bahwa pengiriman dikoordinasikan dengan jadwal konversi sehingga peralatan tersedia saat dibutuhkan. Tidak ada yang dapat menghentikan rencana konversi di jalurnya semudah kegagalan vendor untuk mengirimkan peralatan yang dibutuhkan.

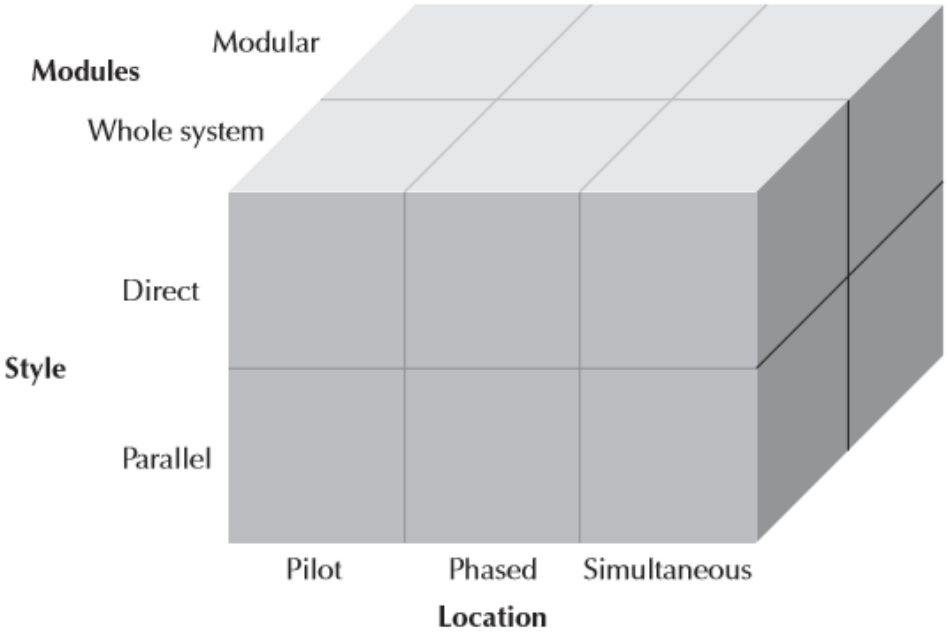
Setelah perangkat keras dipasang, diuji, dan disertifikasi sebagai operasional, langkah kedua adalah memasang perangkat lunak. Ini termasuk sistem yang akan dikembangkan dan, terkadang, perangkat lunak tambahan yang harus diinstal untuk membuat sistem beroperasi. Pada titik ini, sistem biasanya diuji lagi untuk memastikan bahwa sistem beroperasi sesuai rencana.



Gambar 13-2 Elemen Rencana Migrasi

Langkah ketiga adalah mengubah data dari sistem saat in ke sistem yang akan datang. Konversi data biasanya merupakan langkah yang paling rumit secara teknis dalam rencana migrasi. Seringkali, program terpisah harus ditulis untuk mengubah data dari sistem saat in ke format baru yang diperlukan dalam sistem yang akan datang dan menyimpannya dalam file sistem dan database yang akan datang. Proses ini sering diperumit oleh fakta bahwa file dan database dalam sistem yang akan datang tidak sama persis dengan file dan database dalam sistem saat in (misalnya, sistem yang akan datang dapat menggunakan beberapa tabel dalam database untuk menyimpan data pelanggan yang terkandung dalam satu file dalam sistem apa adanya). Rencana pengujian formal selalu diperlukan untuk upaya konversi data (lihat Bab 12).

Konversi dapat dipikirkan dalam tiga dimensi: gaya di mana konversi dilakukan (gaya konversi), lokasi atau kelompok kerja apa yang dikonversi pada jam berapa (lokasi konversi), dan modul sistem apa yang dikonversi pada waktu tertentu (konversi modul). Gambar 13-3 menunjukkan hubungan potensial di antara ketiga dimensi ini.



Gambar 13-3 Strategi Konversi

Gaya Konversi

Gaya konversi adalah cara pengguna beralih antara sistem lama dan baru. Ada dua pendekatan yang berbeda secara mendasar terhadap gaya konversi: konversi langsung dan konversi paralel.

Konversi Langsung (*Direct Converssion*). Dengan konversi langsung (kadang-kadang disebut kalkun dingin, big bang, atau pergantian mendadak), sistem baru langsung menggantikan sistem lama. Sistem baru dihidupkan, dan sistem lama segera dimatikan. Ini adalah pendekatan yang kemungkinan besar akan kami gunakan ketika kami meningkatkan perangkat lunak komersial (mis., Microsoft Word) dari satu versi ke versi lain; kita cukup mulai menggunakan versi baru dan berhenti menggunakan versi lama.

Konversi langsung adalah yang paling sederhana dan paling mudah. Namun, ini adalah yang paling berisiko karena masalah apa pun dengan sistem baru yang lolos dari deteksi selama pengujian dapat sangat mengganggu organisasi.

Konversi Paralel (*Parallel Conversion*). Dengan konversi paralel, sistem baru dioperasikan berdampingan dengan sistem lama; kedua sistem digunakan secara bersamaan. Misalnya, jika sistem akuntansi baru dipasang, organisasi memasukkan data ke dalam sistem lama dan sistem baru dan kemudian dengan hati-hati membandingkan output dari kedua sistem untuk memastikan bahwa sistem baru bekerja dengan benar. Setelah beberapa periode waktu (seringkali dalam satu hingga dua bulan) operasi paralel dan perbandingan yang intens antara kedua sistem, sistem lama dimatikan dan organisasi terus menggunakan sistem baru.

Pendekatan ini lebih mungkin untuk menangkap bug utama dalam sistem baru dan mencegah organisasi menderita masalah besar. Jika masalah ditemukan dalam sistem baru, sistem hanya dimatikan dan diperbaiki dan kemudian proses konversi dimulai lagi. Masalah dengan pendekatan ini adalah biaya tambahan untuk mengoperasikan dua sistem yang menjalankan fungsi yang sama.

Lokasi Konversi

Lokasi konversi mengacu pada bagian organisasi yang dikonversi ketika konversi terjadi. Seringkali, bagian dari organisasi secara fisik terletak di kantor yang berbeda (misalnya, Toronto, Atlanta, Los Angeles). Dalam kasus lain, lokasi mengacu pada unit organisasi berbeda yang terletak di berbagai bagian kompleks kantor yang sama (misalnya, entri pesanan, pengiriman, pembelian). Setidaknya ada tiga pendekatan yang berbeda secara mendasar untuk memilih cara lokasi organisasi yang berbeda dikonversi: konversi percontohan, konversi bertahap, dan konversi simultan.

Konversi Percontohan (*Pilot Conversion)* Dengan konversi percontohan, satu atau lebih lokasi atau unit atau kelompok kerja dalam suatu lokasi dipilih untuk dikonversi terlebih dahulu sebagai bagian dari uji percontohan. Lokasi yang berpartisipasi dalam uji coba dikonversi (menggunakan konversi langsung atau paralel). Jika sistem lulus uji coba, maka sistem dipasang di lokasi yang tersisa (sekali lagi menggunakan konversi langsung atau paralel).

Konversi percontohan memiliki keuntungan menyediakan tingkat pengujian tambahan sebelum sistem diterapkan secara luas di seluruh organisasi, sehingga masalah apa pun dengan sistem hanya memengaruhi lokasi percontohan. Namun, jenis konversi ini jelas membutuhkan lebih banyak waktu sebelum sistem dipasang di semua lokasi organisasi. Juga, ini berarti bahwa unit organisasi yang berbeda menggunakan versi sistem dan proses bisnis yang berbeda, yang dapat mempersulit mereka untuk bertukar data.

Konversi Bertahap (*Phased Conversion*). Dengan konversi bertahap, sistem dipasang secara berurutan di lokasi yang berbeda. Kumpulan lokasi pertama dikonversi, lalu kumpulan kedua, lalu kumpulan ketiga, dan seterusnya, hingga semua lokasi dikonversi. Kadang-kadang ada penundaan yang disengaja antara set yang berbeda (setidaknya antara yang pertama dan yang kedua), sehingga masalah apa pun dengan sistem terdeteksi sebelum terlalu banyak organisasi terpengaruh. Dalam kasus lain, set dikonversi kembali ke belakang sehingga segera setelah mereka yang mengonversi satu lokasi selesai, tim proyek pindah ke yang berikutnya dan melanjutkan konversi.

Konversi bertahap memiliki kelebihan dan kekurangan yang sama dengan konversi percontohan. Selain itu, ini berarti lebih sedikit orang yang diperlukan untuk melakukan konversi aktual (dan pelatihan pengguna terkait) dibandingkan jika semua lokasi dikonversi sekaligus.

Konversi Serentak *(Simultaneous Conversion)*. Konversi simultan, seperti namanya, berarti bahwa semua lokasi dikonversi pada waktu yang sama. Sistem baru dipasang dan disiapkan di semua lokasi; pada waktu yang telah ditentukan, semua pengguna mulai menggunakan sistem baru. Konversi simultan sering digunakan dengan konversi langsung, tetapi juga dapat digunakan dengan konversi paralel.

Konversi simultan menghilangkan masalah dengan memiliki unit organisasi yang berbeda menggunakan sistem dan proses yang berbeda. Namun, itu juga berarti bahwa organisasi harus memiliki staf yang cukup untuk melakukan konversi dan melatih pengguna di semua lokasi secara bersamaan.

Modul Konversi

Meskipun wajar untuk mengasumsikan bahwa sistem biasanya dipasang secara keseluruhan, hal ini tidak selalu terjadi.

Konversi Seluruh Sistem (*Whole-System Conversion*). Konversi seluruh sistem, di mana seluruh sistem diinstal pada satu waktu, adalah yang paling umum. Sederhana dan paling mudah dipahami. Namun, jika sistemnya besar dan/atau sangat kompleks (misalnya, sistem perencanaan sumber daya perusahaan seperti SAP atau PeopleSoft), seluruh sistem dapat terbukti terlalu sulit untuk dipelajari pengguna dalam satu langkah konversi.

Konversi Modular (*Modular Conversion)*. Ketika modul dalam suatu sistem terpisah dan berbeda, organisasi terkadang memilih untuk mengonversi ke sistem baru satu modul pada satu waktu—yaitu, menggunakan konversi modular. Konversi modular memerlukan perhatian khusus dalam mengembangkan sistem (dan biasanya menambah biaya tambahan). Setiap modul harus ditulis untuk bekerja dengan sistem lama dan baru atau pembungkus objek (lihat Bab 7) harus digunakan untuk merangkum sistem lama dari yang baru. Ketika modul terintegrasi, ini sangat menantang dan karena itu jarang dilakukan. Namun, ketika hanya ada hubungan longgar antar modul, konversi modul menjadi lebih mudah. Misalnya, pertimbangkan konversi dari versi lama Microsoft Office ke versi baru. Relatif mudah untuk mengonversi dari versi lama Word ke versi baru tanpa secara bersamaan harus mengubah dari versi lama ke versi baru Microsoft Excel.

Konversi modular mengurangi jumlah pelatihan yang diperlukan untuk mulai menggunakan sistem baru. Pengguna hanya membutuhkan pelatihan dalam modul baru yang diterapkan. Namun, konversi modular memang memakan waktu lebih lama dan memiliki lebih banyak langkah daripada proses keseluruhan sistem.

Memilih Strategi Konversi yang Tepat

Masing-masing dari tiga dimensi pada Gambar 13-3 adalah independen, sehingga strategi konversi dapat dikembangkan agar sesuai dengan salah satu kotak dalam gambar ini. Kotak yang berbeda juga dapat dicampur dan dicocokkan menjadi satu strategi konversi. Misalnya, satu pendekatan yang umum digunakan adalah memulai dengan konversi percontohan dari keseluruhan sistem menggunakan konversi paralel di beberapa lokasi pengujian. Setelah sistem lulus uji coba di lokasi ini, sistem kemudian dipasang di lokasi lainnya menggunakan konversi bertahap dengan pengalihan langsung. Ada tiga faktor penting yang perlu dipertimbangkan dalam memilih strategi konversi: risiko, biaya, dan waktu yang dibutuhkan (Gambar 13-4).

Resiko. Setelah sistem melewati serangkaian pengujian unit, sistem, integrasi, dan penerimaan yang ketat, sistem harus bebas bug . . . mungkin. Karena manusia membuat kesalahan, tidak ada yang dibangun oleh manusia yang pernah sempurna. Bahkan setelah

semua tes ini, mungkin masih ada beberapa bug yang belum ditemukan. Proses konversi menyediakan satu langkah terakhir untuk menangkap bug ini sebelum sistem ditayangkan dan bug memiliki peluang untuk menyebabkan masalah.

Konversi paralel kurang berisiko daripada konversi langsung karena memiliki peluang lebih besar untuk mendeteksi bug yang belum ditemukan dalam pengujian. Demikian juga, konversi percontohan kurang berisiko daripada konversi bertahap atau konversi simultan karena jika bug benar-benar terjadi, mereka terjadi di lokasi uji coba yang stafnya sadar bahwa mereka mungkin menemukan bug. Karena potensi bug memengaruhi lebih sedikit pengguna, risikonya lebih kecil. Demikian juga, mengonversi beberapa modul sekaligus menurunkan kemungkinan bug karena kemungkinan besar ada bug di seluruh sistem daripada di modul mana pun.

Seberapa penting risikonya tergantung pada sistem yang diimplementasikan—kombinasi dari kemungkinan bahwa bug tetap tidak terdeteksi dalam sistem dan potensi biaya dari bug yang tidak terdeteksi tersebut. Jika sistem memang telah mengalami pengujian metodis yang ekstensif, termasuk pengujian alfa dan beta, maka kemungkinan bug yang tidak terdeteksi lebih rendah daripada jika pengujiannya kurang ketat. Namun, mungkin masih ada kesalahan yang dibuat dalam proses analisis, sehingga meskipun mungkin tidak ada bug perangkat lunak, perangkat lunak mungkin gagal memenuhi kebutuhan bisnis dengan benar.

	Conversion Style		Conversion Location			Conversion Modules	
Characteristic	Direct Conversion	Parallel Conversion	Pilot Conversion	Phased Conversion	Simultaneous Conversion	Whole-System Conversion	Modular Conversion
Risk	High	Low	Low	Medium	High	High	Medium
Cost	Low	High	Medium	Medium	High	Medium	High
Time	Short	Long	Medium	Long	Short	Short	Long

Gambar 13-4 Karakteristik Strategi Konversi

Menilai biaya bug itu menantang, tetapi sebagian besar analis dan manajer senior dapat membuat perkiraan yang masuk akal tentang biaya relatif dari sebuah bug. Misalnya, biaya bug dalam program perdagangan pasar saham otomatis atau mesin jantung-paru yang membuat seseorang tetap hidup kemungkinan akan jauh lebih besar daripada bug dalam permainan komputer atau program pengolah kata. Oleh karena itu, risiko kemungkinan menjadi faktor yang sangat penting dalam proses konversi jika sistem belum diuji secara menyeluruh seperti yang mungkin terjadi atau jika biaya bug tinggi. Jika sistem telah diuji secara menyeluruh atau biaya bug tidak terlalu tinggi, maka risiko menjadi kurang penting dalam keputusan konversi.

Biaya. Seperti yang diharapkan, strategi konversi yang berbeda memiliki biaya yang berbeda. Biaya ini dapat mencakup hal-hal seperti gaji untuk orang yang bekerja dengan sistem (misalnya, pengguna, pelatih, administrator sistem, konsultan eksternal), biaya perjalanan, biaya operasi, biaya komunikasi, dan sewa perangkat keras. Konversi paralel lebih mahal daripada pengalihan langsung karena memerlukan dua sistem (yang lama dan yang baru) dioperasikan pada waktu yang sama. Karyawan kemudian harus melakukan dua kali pekerjaan biasa karena mereka harus memasukkan data yang sama ke dalam sistem lama dan baru. Konversi paralel juga mengharuskan hasil dari kedua sistem diperiksa silang sepenuhnya untuk memastikan tidak ada perbedaan di antara keduanya, yang memerlukan waktu dan biaya tambahan.

Konversi percontohan dan konversi bertahap memiliki biaya yang agak mirip. Konversi simultan memiliki biaya lebih tinggi karena lebih banyak staf diperlukan untuk

mendukung semua lokasi karena mereka secara bersamaan beralih dari sistem lama ke sistem baru. Konversi modular lebih mahal daripada konversi seluruh sistem karena membutuhkan lebih banyak pemrograman. Sistem lama harus diperbarui untuk bekerja dengan modul yang dipilih dalam sistem baru, dan modul dalam sistem baru harus diprogram untuk bekerja dengan modul yang dipilih di sistem lama dan baru.

Waktu. Faktor terakhir adalah jumlah waktu yang dibutuhkan untuk mengubah antara sistem lama dan baru. Konversi langsung adalah yang tercepat karena langsung. Konversi paralel membutuhkan waktu lebih lama karena keuntungan penuh dari sistem baru tidak tersedia sampai sistem lama dimatikan. Konversi simultan paling cepat karena semua lokasi dikonversi pada waktu yang sama. Konversi bertahap biasanya memakan waktu lebih lama daripada konversi percontohan karena setelah uji coba selesai, semua lokasi yang tersisa biasanya (tetapi tidak selalu) dikonversi secara bersamaan. Konversi bertahap berlangsung dalam gelombang, seringkali memerlukan beberapa bulan sebelum semua lokasi dikonversi. Demikian pula, konversi modular membutuhkan waktu lebih lama daripada konversi seluruh sistem karena model diperkenalkan satu demi satu.

13.5 PERUBAHAN MANAJEMEN

Dalam konteks proyek pengembangan sistem, manajemen perubahan adalah proses membantu orang untuk mengadopsi dan beradaptasi dengan sistem yang akan datang dan proses kerja yang menyertainya tanpa tekanan yang tidak semestinya. Ada tiga peran utama dalam setiap perubahan organisasi besar. Yang pertama adalah sponsor perubahan—orang yang menginginkan perubahan. Orang ini adalah sponsor bisnis yang pertama kali mengajukan permintaan untuk sistem baru (lihat Bab 2). Biasanya, sponsor adalah manajer senior dari bagian organisasi yang harus mengadopsi dan menggunakan sistem baru. Sangat penting bahwa sponsor aktif dalam proses manajemen perubahan karena perubahan yang jelas didorong oleh sponsor, bukan oleh tim proyek atau organisasi SI, memiliki legitimasi yang lebih besar. Sponsor memiliki otoritas manajemen langsung atas mereka yang mengadopsi sistem.

Peran kedua adalah sebagai agen perubahan—orang yang memimpin upaya perubahan. Agen perubahan, yang bertanggung jawab untuk merencanakan dan mengimplementasikan perubahan, biasanya adalah seseorang di luar unit bisnis yang mengadopsi sistem dan oleh karena itu tidak memiliki otoritas manajemen langsung atas calon pengadopsi. Karena agen perubahan adalah orang luar, kredibilitasnya lebih rendah daripada sponsor dan anggota unit bisnis lainnya. Lagi pula, setelah sistem diinstal, agen perubahan biasanya pergi dan dengan demikian tidak memiliki dampak berkelanjutan.

Peran ketiga adalah pengadopsi potensial, atau target perubahan—orang-orang yang benar-benar harus berubah. Ini adalah orang-orang untuk siapa sistem baru dirancang dan yang pada akhirnya akan memilih untuk menggunakan atau tidak menggunakan sistem.

Pada hari-hari awal komputasi, banyak tim proyek hanya berasumsi bahwa pekerjaan mereka berakhir ketika sistem lama diubah ke sistem baru pada tingkat teknis. Filosofinya adalah "bangun dan mereka akan datang." Sayangnya, itu hanya terjadi di film. Resistensi terhadap perubahan umum terjadi di sebagian besar organisasi. Oleh karena itu, rencana manajemen perubahan merupakan bagian penting dari keseluruhan rencana instalasi yang menyatukan langkah-langkah utama dalam proses manajemen perubahan. Perubahan yang berhasil mengharuskan orang mau mengadopsi perubahan dan mampu mengadopsi perubahan. Rencana manajemen perubahan memiliki empat langkah dasar: merevisi kebijakan manajemen, menilai model biaya dan manfaat dari pengadopsi potensial, memotivasi adopsi, dan memungkinkan orang untuk mengadopsi melalui pelatihan (lihat Gambar 13-2). Namun,

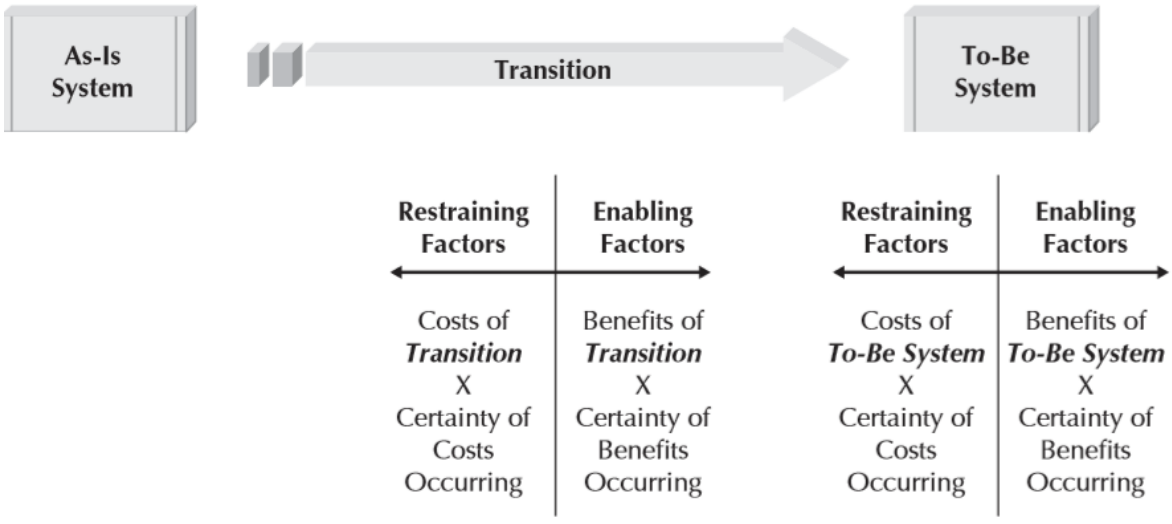
sebelum kita dapat membahas rencana manajemen perubahan, pertama-tama kita harus memahami mengapa orang menolak perubahan.

Memahami Resistensi terhadap Perubahan

Orang-orang menolak perubahan—bahkan perubahan menjadi lebih baik—karena alasan yang sangat rasional. Apa yang baik bagi organisasi belum tentu baik bagi orang-orang yang bekerja di sana. Sebagai contoh, perhatikan seorang pegawai pemroses pesanan yang dulunya menerima pesanan untuk dikirim dalam dokumen pengiriman kertas tetapi sekarang menggunakan komputer untuk menerima informasi yang sama. Daripada mengetik label pengiriman dengan mesin tik, petugas sekarang mengklik tombol cetak di komputer dan label diproduksi secara otomatis. Petugas sekarang dapat mengirimkan lebih banyak pesanan setiap hari, yang merupakan keuntungan nyata bagi organisasi. Petugas, bagaimanapun, mungkin tidak terlalu peduli berapa banyak paket yang dikirim. Gajinya tidak berubah; itu hanya pertanyaan yang lebih disukai petugas, komputer atau mesin tik. Belajar menggunakan sistem dan proses kerja baru—walaupun perubahannya kecil—membutuhkan lebih banyak usaha daripada terus menggunakan sistem dan proses kerja yang sudah ada dan dipahami dengan baik.

Jadi mengapa orang menerima perubahan? Sederhananya, setiap perubahan memiliki serangkaian biaya dan manfaat yang terkait dengannya. Jika manfaat menerima perubahan lebih besar daripada biaya perubahan, maka orang berubah. Terkadang manfaat perubahan adalah menghindari rasa sakit yang mungkin dialami jika perubahan tidak diadopsi (misalnya jika Anda tidak berubah, Anda dipecat, jadi salah satu manfaat mengadopsi perubahan adalah Anda masih memiliki pekerjaan.).

Secara umum, ketika orang diberi kesempatan untuk berubah, mereka melakukan analisis biaya-manfaat (kadang secara sadar, kadang tidak sadar) dan memutuskan sejauh mana mereka akan menerima dan mengadopsi perubahan. Mereka mengidentifikasi biaya dan manfaat dari sistem dan memutuskan apakah perubahan itu bermanfaat. Namun, tidak sesederhana itu, karena sebagian besar biaya dan manfaat tidak pasti. Ada beberapa ketidakpastian apakah manfaat atau biaya tertentu akan benar-benar terjadi; sehingga baik biaya maupun manfaat dari sistem baru perlu ditimbang dengan tingkat kepastian yang terkait dengannya (Gambar 13-5). Sayangnya, kebanyakan manusia cenderung melebih-lebihkan kemungkinan biaya dan meremehkan kemungkinan manfaat.



Gambar 13-5 Biaya dan Manfaat Perubahan

Ada juga biaya dan, terkadang, manfaat yang terkait dengan proses transisi itu sendiri. Sebagai contoh, misalkan kita menemukan rumah atau apartemen yang lebih bagus daripada rumah kita saat ini. Bahkan jika kami lebih menyukainya, kami mungkin memutuskan untuk tidak pindah hanya karena biaya pindahnya lebih besar daripada manfaat dari rumah atau apartemen baru itu sendiri. Demikian juga, mengadopsi sistem komputer baru mungkin mengharuskan kita untuk mempelajari keterampilan baru, yang dapat dilihat sebagai biaya bagi sebagian orang atau sebagai manfaat bagi orang lain, jika mereka merasa bahwa keterampilan itu entah bagaimana akan memberikan manfaat lain di luar penggunaan sistem itu sendiri. . Sekali lagi, setiap biaya dan manfaat dari proses transisi harus ditimbang dengan kepastian yang akan terjadi (lihat Gambar 13-5).

Secara bersama-sama, kedua set biaya dan manfaat ini (dan kepastian relatifnya) mempengaruhi penerimaan perubahan atau penolakan terhadap perubahan yang dihadapi tim proyek saat memasang sistem baru dalam organisasi. Langkah pertama dalam manajemen perubahan adalah memahami faktor-faktor yang menghambat perubahan—faktor-faktor yang memengaruhi persepsi biaya dan manfaat serta kepastian bahwa semua itu akan dihasilkan oleh sistem baru. Sangat penting untuk memahami bahwa biaya nyata dan manfaat nyata jauh lebih penting daripada biaya yang dirasakan dan manfaat yang dirasakan. Orang-orang bertindak berdasarkan apa yang mereka yakini benar, bukan berdasarkan apa yang benar. Dengan demikian, setiap pemahaman tentang bagaimana memotivasi perubahan harus dikembangkan dari sudut pandang orang-orang yang diharapkan untuk berubah, bukan dari sudut pandang mereka yang memimpin perubahan.

Merevisi Kebijakan Manajemen

Langkah besar pertama dalam rencana manajemen perubahan adalah mengubah kebijakan manajemen yang dirancang untuk sistem saat in menjadi kebijakan manajemen baru yang dirancang untuk mendukung sistem yang akan datang. Kebijakan manajemen memberikan tujuan, menentukan bagaimana proses kerja harus dilakukan, dan menentukan bagaimana anggota organisasi dihargai. Tidak ada sistem komputer yang akan berhasil diadopsi kecuali kebijakan manajemen mendukung penerapannya. Banyak sistem komputer baru membawa perubahan pada proses bisnis; mereka memungkinkan cara kerja baru. Kecuali jika kebijakan yang memberikan aturan dan penghargaan untuk proses tersebut direvisi untuk mencerminkan peluang baru yang diizinkan oleh sistem, pengadopsi potensial tidak dapat dengan mudah menggunakannya.

Manajemen memiliki tiga alat dasar untuk menyusun proses kerja dalam organisasi.8 Yang pertama adalah prosedur operasi standar/*standard operating procedures* (SOP) yang menjadi rutinitas kebiasaan bagaimana pekerjaan dilakukan. SOP tersebut bersifat formal dan informal. SOP formal mendefinisikan perilaku yang tepat. SOP informal adalah norma yang telah berkembang dari waktu ke waktu tentang bagaimana proses sebenarnya dilakukan. Manajemen harus memastikan bahwa SOP formal direvisi agar sesuai dengan sistem yang akan dibuat. SOP informal kemudian akan berkembang untuk menyempurnakan dan mengisi rincian yang tidak ada dalam SOP formal.

Aspek kedua dari kebijakan manajemen adalah mendefinisikan bagaimana orang memberikan makna pada peristiwa. Apa artinya "menjadi sukses" atau "melakukan pekerjaan dengan baik"? Kebijakan membantu orang memahami makna dengan mendefinisikan pengukuran dan penghargaan. Pengukuran secara eksplisit mendefinisikan makna karena memberikan bukti yang jelas dan konkret tentang apa yang penting bagi organisasi. Imbalan

memperkuat pengukuran karena "apa yang diukur akan dilakukan" (pepatah yang terlalu sering digunakan tetapi akurat). Pengukuran harus dirancang dengan hati-hati untuk memotivasi perilaku yang diinginkan.

Aspek ketiga dari kebijakan manajemen adalah alokasi sumber daya. Manajer dapat memiliki dampak yang jelas dan langsung pada perilaku dengan mengalokasikan sumber daya. Mereka dapat mengarahkan dana dan staf dari satu proyek ke proyek lainnya, membuat infrastruktur yang mendukung sistem baru, dan berinvestasi dalam program pelatihan. Masing-masing aktivitas ini memiliki efek langsung dan simbolis. Efek langsung berasal dari realokasi sumber daya yang sebenarnya. Efek simbolik menunjukkan bahwa manajemen serius dengan niatnya. Ada sedikit ketidakpastian tentang komitmen jangka panjang manajemen untuk sistem baru ketika pengadopsi potensial melihat sumber daya berkomitmen untuk mendukungnya.

Menilai Biaya dan Manfaat

Langkah selanjutnya dalam mengembangkan rencana manajemen perubahan adalah mengembangkan dua daftar biaya dan manfaat yang jelas dan ringkas yang disediakan oleh sistem baru (dan transisinya) dibandingkan dengan sistem apa adanya. Daftar pertama dikembangkan dari perspektif organisasi, yang harus mengalir dengan mudah dari kasus bisnis yang dikembangkan selama studi kelayakan dan disempurnakan selama umur proyek (lihat Bab 2). Kumpulan biaya dan manfaat organisasi ini harus didistribusikan secara luas sehingga setiap orang yang diharapkan untuk mengadopsi sistem baru harus memahami dengan jelas mengapa sistem baru itu berharga bagi organisasi.

Daftar biaya dan manfaat kedua dikembangkan dari sudut pandang pengadopsi potensial yang berbeda yang diharapkan berubah, atau pemegang kepentingan dalam perubahan. Misalnya, satu set pengadopsi potensial mungkin adalah karyawan garis depan, yang lain mungkin penyelia lini pertama, dan yang lain mungkin manajemen menengah. Masing-masing pengadopsi potensial ini, atau pemegang kepentingan, mungkin memiliki serangkaian biaya dan manfaat yang berbeda terkait dengan perubahan—biaya dan manfaat yang dapat sangat berbeda dari biaya dan manfaat organisasi. Dalam beberapa situasi, serikat pekerja mungkin menjadi pemegang kepentingan utama yang dapat membuat atau menghancurkan perubahan yang berhasil.

Banyak analis sistem secara alami berasumsi bahwa karyawan garis depan adalah orang-orang yang set biaya dan manfaat yang paling mungkin menyimpang dari organisasi dan dengan demikian adalah orang-orang yang paling menolak perubahan. Namun, mereka biasanya menanggung beban masalah dengan sistem saat ini. Ketika masalah terjadi, mereka sering mengalaminya secara langsung. Manajer menengah dan supervisor lini pertama adalah yang paling mungkin memiliki serangkaian biaya dan manfaat yang berbeda dan, oleh karena itu, menolak perubahan karena sistem komputer baru sering mengubah seberapa besar kekuatan yang mereka miliki. Misalnya, sistem komputer baru dapat meningkatkan kontrol organisasi atas proses kerja (manfaat bagi organisasi) tetapi mengurangi kekuatan pengambilan keputusan manajemen menengah (biaya yang jelas bagi manajer menengah).

Analisis biaya dan manfaat untuk setiap kelompok pengadopsi potensial, atau pemegang kepentingan, akan membantu menentukan mereka yang kemungkinan besar akan mendukung perubahan dan mereka yang mungkin menolak perubahan. Tantangannya pada titik ini adalah mencoba mengubah keseimbangan biaya dan manfaat bagi mereka yang diharapkan untuk menolak perubahan sehingga mereka mendukungnya (atau setidaknya tidak secara aktif menolaknya). Analisis ini dapat mengungkap beberapa masalah serius yang berpotensi menghalangi keberhasilan penerapan sistem. Mungkin perlu untuk mengkaji ulang kebijakan manajemen dan membuat perubahan signifikan untuk memastikan bahwa

keseimbangan biaya dan manfaat sedemikian rupa sehingga pengadopsi potensial yang penting termotivasi untuk mengadopsi sistem.

Gambar 13-6 merangkum beberapa faktor yang penting untuk keberhasilan perubahan. Alasan pertama dan terpenting adalah alasan pribadi yang kuat untuk berubah. Semua perubahan dilakukan oleh individu, bukan organisasi. Jika ada alasan kuat bagi kelompok utama dari masing-masing pemegang kepentingan untuk menginginkan perubahan, maka perubahan tersebut kemungkinan besar akan berhasil. Faktor-faktor seperti kenaikan gaji, pengurangan ketidaknyamanan, dan—bergantung pada individu—peluang untuk promosi dan pengembangan pribadi dapat menjadi motivator penting. Namun, jika perubahan membuat keterampilan saat ini kurang berharga, individu mungkin menolak perubahan karena mereka telah menginvestasikan banyak waktu dan energi dalam memperoleh keterampilan tersebut, dan apa pun yang mengurangi keterampilan tersebut dapat dianggap mengurangi individu (karena keterampilan penting membawa rasa hormat. dan kekuasaan).

Juga harus ada alasan kuat bagi organisasi untuk membutuhkan perubahan; jika tidak, individu menjadi skeptis bahwa perubahan itu penting dan kurang yakin bahwa perubahan itu akan benar-benar terjadi. Mungkin organisasi yang paling sulit untuk diubah adalah organisasi yang telah berhasil karena individu menjadi percaya bahwa apa yang berhasil di masa lalu akan terus berhasil. Sebaliknya, dalam organisasi yang berada di ambang kebangkrutan, lebih mudah untuk meyakinkan individu bahwa perubahan diperlukan. Komitmen dan dukungan dari sponsor bisnis yang kredibel dan manajemen puncak juga penting dalam meningkatkan kepastian bahwa perubahan akan terjadi.

Kemungkinan perubahan yang berhasil meningkat ketika biaya transisi ke individu yang harus berubah rendah. Kebutuhan akan keterampilan baru yang sangat berbeda atau gangguan dalam operasi dan kebiasaan kerja dapat menciptakan resistensi. Rencana migrasi yang jelas yang dikembangkan oleh agen perubahan yang kredibel yang mendapat dukungan dari sponsor bisnis merupakan faktor penting dalam meningkatkan kepastian tentang biaya proses transisi.

Memotivasi Adopsi

Satu-satunya faktor terpenting dalam memotivasi perubahan adalah memberikan bukti yang jelas dan meyakinkan tentang perlunya perubahan. Sederhananya, setiap orang yang diharapkan untuk mengadopsi perubahan harus yakin bahwa manfaat dari sistem yang akan datang lebih besar daripada biaya perubahan.

Ada dua strategi dasar untuk memotivasi adopsi: informasional dan politis. Kedua strategi tersebut sering digunakan secara bersamaan. Dengan strategi informasi, tujuannya adalah untuk meyakinkan calon pengadopsi bahwa perubahan itu menjadi lebih baik. Strategi ini bekerja ketika set biaya-manfaat dari pengadopsi target memiliki lebih banyak manfaat daripada biaya. Dengan kata lain, memang ada alasan yang jelas bagi para calon pengadopsi untuk menyambut perubahan tersebut.

Dengan menggunakan pendekatan ini, tim proyek memberikan bukti yang jelas dan meyakinkan tentang biaya dan manfaat pindah ke sistem yang akan datang. Tim proyek menulis memo dan mengembangkan presentasi yang menguraikan biaya dan manfaat mengadopsi sistem dari perspektif organisasi dan dari perspektif kelompok sasaran pengadopsi potensial. Informasi ini disebarluaskan ke seluruh kelompok sasaran, seperti iklan atau kampanye hubungan masyarakat. Ini harus menekankan manfaat dan meningkatkan kepastian di benak calon pengadopsi bahwa manfaat ini benar-benar akan tercapai. Dalam

pengalaman kami, selalu lebih mudah untuk menjual obat penghilang rasa sakit daripada vitamin; yaitu, lebih mudah untuk meyakinkan calon pengadopsi bahwa sistem baru akan menghilangkan masalah besar (atau sumber masalah lain) daripada memberikan manfaat baru (misalnya, meningkatkan penjualan). Oleh karena itu, kampanye informasi lebih mungkin berhasil jika menekankan pengurangan atau penghapusan masalah daripada berfokus pada penyediaan peluang baru.

	Faktor	Contoh	Efek	Tindakan yang harus dilakukan
Manfaat sistem yang akan datang	Alasan pribadi yang menarik untuk perubahan	Peningkatan gaji, lebih sedikit aspek yang tidak menyenangkan, kesempatan untuk promosi, sebagian besar keterampilan yang ada tetap berharga.	Jika sistem baru memberikan manfaat pribadi yang jelas bagi mereka yang harus mengadopsinya, mereka cenderung menerima perubahan tersebut.	Lakukan analisis biaya-manfaat dari sudut pandang pemegang kepentingan, buat perubahan jika diperlukan, dan secara aktif mempromosikan manfaat.
Kepastian manfaat	Alasan organisasi yang menarik untuk perubahan	Risiko kebangkrutan, akuisisi, peraturan pemerintah	Jika pengadopsi tidak mengerti mengapa organisasi menerapkan perubahan, mereka kurang yakin bahwa perubahan akan terjadi.	Lakukan analisis biaya-manfaat dari sudut pandang organisasi dan luncurkan kampanye informasi yang gencar untuk menjelaskan hasilnya kepada semua orang.
	Dukungan manajemen puncak yang ditunjukkan	Keterlibatan aktif, sering disebutkan dalam pidato	Jika manajemen puncak tidak terlihat secara aktif mendukung perubahan, maka kemungkinan kecil perubahan itu akan terjadi.	Mendorong manajemen puncak untuk berpartisipasi dalam kampanye informasi.
	Sponsor bisnis yang berkomitmen dan terlibat	Keterlibatan aktif, sering mengunjungi pengguna dan tim proyek, memperjuangkan,	Jika sponsor bisnis (manajer fungsional yang memprakarsai proyek) tidak terlihat secara aktif mendukung perubahan, ada sedikit kepastian bahwa perubahan akan terjadi.	Dorong sponsor bisnis untuk berpartisipasi dalam kampanye informasi dan berperan aktif dalam rencana manajemen perubahan.

	Manajemen puncak dan sponsor bisnis yang kredibel	Manajemen dan sponsor yang melakukan apa yang mereka katakan alih-alih menjadi anggota klub "mode manajemen bulan ini"	Jika sponsor bisnis dan manajemen puncak memiliki kredibilitas di mata pengadopsi, kepastian manfaat yang diklaim lebih tinggi.	Memastikan bahwa sponsor bisnis dan/atau manajemen puncak memiliki kredibilitas sehingga keterlibatan tersebut akan membantu; jika tidak ada kredibilitas, keterlibatan akan berdampak kecil.
Biaya transisi	Biaya perubahan pribadi yang rendah	Sedikit keterampilan baru yang dibutuhkan	Biaya perubahan tidak ditanggung secara merata oleh semua pemegang kepentingan; biaya cenderung lebih tinggi untuk beberapa.	Lakukan analisis biaya-manfaat dari sudut pandang pemegang kepentingan, buat perubahan jika diperlukan, dan secara aktif mempromosikan biaya rendah.
Kepastian biaya	Rencana yang jelas untuk perubahan	Tanggal dan instruksi yang jelas untuk perubahan, harapan yang jelas	Jika ada rencana migrasi yang jelas, kemungkinan akan menurunkan biaya transisi yang dirasakan.	Publikasikan rencana migrasi.
	Agen perubahan yang kredibel	Pengalaman sebelumnya dengan perubahan, melakukan apa yang dia janjikan untuk dilakukan	Jika agen perubahan memiliki kredibilitas di mata pengadopsi, kepastian biaya yang diklaim lebih tinggi.	Jika agen perubahan tidak kredibel, maka perubahan akan sulit.
	Mandat yang jelas untuk agen perubahan dari sponsor	Dukungan terbuka untuk agen perubahan ketika terjadi ketidaksepakatan	Jika agen perubahan memiliki mandat yang jelas dari sponsor bisnis, kepastian biaya yang diklaim lebih tinggi.	Sponsor bisnis harus secara aktif menunjukkan dukungan untuk agen perubahan.

Gambar 13-6 Faktor Utama dalam Keberhasilan Perubahan

Strategi lain untuk memotivasi perubahan adalah strategi politik. Dengan strategi politik, kekuatan organisasi, bukan informasi, digunakan untuk memotivasi perubahan. Pendekatan ini sering digunakan ketika set biaya-manfaat dari pengadopsi target memiliki lebih banyak biaya daripada manfaat. Dengan kata lain, meskipun perubahan mungkin

menguntungkan organisasi, tidak ada alasan bagi calon pengadopsi untuk menyambut perubahan tersebut.

Sponsor	Agen Perubahan	Pengadopsi Potensial
Sponsor menginginkan perubahan terjadi.	Agen perubahan memimpin upaya perubahan.	Pengadopsi potensial adalah orang-orang yang harus berubah. 20-30 persen adalah pengadopsi siap. 20-30 persen adalah pengadopsi yang resisten. 40-60 persen adalah pengadopsi yang enggan.

Gambar 13-7 Aktor dalam Proses Manajemen Perubahan

Strategi politik biasanya di luar kendali tim proyek. Hal ini membutuhkan seseorang dalam organisasi yang memegang kekuasaan yang sah atas kelompok sasaran untuk mempengaruhi kelompok untuk mengadopsi perubahan. Ini dapat dilakukan dengan cara koersif (misalnya, mengadopsi sistem atau Anda dipecat) atau dengan cara yang dinegosiasikan, di mana kelompok sasaran memperoleh manfaat dengan cara lain yang terkait dengan penerapan sistem (misalnya, menghubungkan sistem adopsi untuk meningkatkan peluang pelatihan). Kebijakan manajemen dapat memainkan peran utama dalam strategi politik dengan menghubungkan gaji dengan perilaku tertentu yang diinginkan dengan sistem baru.

Secara umum, untuk setiap perubahan yang memiliki manfaat organisasi yang sebenarnya, sekitar 20 hingga 30 persen pengadopsi potensial akan menjadi pengadopsi siap. Mereka mengenali manfaat, dengan cepat mengadopsi sistem, dan menjadi pendukung sistem. 20 hingga 30 persen lainnya adalah pengadopsi yang resisten. Mereka hanya menolak untuk menerima perubahan dan mereka melawannya, baik karena sistem baru memiliki biaya lebih dari keuntungan bagi mereka secara pribadi atau karena mereka menempatkan biaya tinggi pada proses transisi itu sendiri sehingga tidak ada jumlah manfaat dari sistem baru yang dapat melebihi mengubah biaya. Sisanya 40 hingga 60 persen adalah pengadopsi yang enggan. Mereka cenderung apatis dan akan mengikuti arus untuk mendukung atau menolak sistem, tergantung pada bagaimana proyek berkembang dan bagaimana rekan kerja mereka bereaksi terhadap sistem. Gambar 13-7 mengilustrasikan aktor yang terlibat dalam proses manajemen perubahan.

Tujuan dari manajemen perubahan adalah untuk secara aktif mendukung dan mendorong pengadopsi yang siap dan membantu mereka memenangkan pengadopsi yang enggan. Biasanya hanya ada sedikit yang dapat dilakukan tentang pengadopsi yang resisten karena set biaya dan manfaat mereka mungkin berbeda dari organisasi. Kecuali ada langkah-langkah sederhana yang dapat diambil untuk menyeimbangkan kembali biaya dan manfaat mereka atau organisasi memilih untuk mengadopsi strategi politik yang kuat, seringkali yang terbaik adalah mengabaikan minoritas kecil pengadopsi yang resisten ini dan fokus pada sebagian besar pengadopsi yang siap dan enggan.

Mengaktifkan Adopsi: Pelatihan

Pengadopsi potensial mungkin ingin mengadopsi perubahan, tetapi kecuali mereka mampu mengadopsinya, mereka tidak akan melakukannya. Pelatihan yang cermat memungkinkan adopsi dengan menyediakan keterampilan yang dibutuhkan untuk mengadopsi perubahan. Pelatihan mungkin merupakan bagian yang paling jelas dari setiap

inisiatif manajemen perubahan. Bagaimana sebuah organisasi dapat mengharapkan anggota stafnya untuk mengadopsi sistem baru jika mereka tidak dilatih? Namun, kami telah menemukan bahwa pelatihan adalah salah satu bagian proses yang paling sering diabaikan. Banyak organisasi dan manajer proyek hanya mengharapkan pengadopsi potensial untuk menemukan sistem yang mudah dipelajari. Karena sistemnya dianggap sangat sederhana, maka calon pengadopsi seharusnya dapat belajar dengan sedikit usaha. Sayangnya, ini biasanya asumsi yang terlalu optimis.

Setiap sistem baru membutuhkan keterampilan baru, baik karena proses kerja dasar telah berubah atau karena sistem komputer yang digunakan untuk mendukung proses tersebut berbeda. Semakin radikal perubahan proses bisnis, semakin penting untuk memastikan organisasi memiliki keterampilan baru yang diperlukan untuk mengoperasikan proses bisnis baru dan mendukung sistem informasi. Secara umum, ada tiga cara untuk mendapatkan keterampilan baru ini. Salah satunya adalah merekrut karyawan baru yang memiliki keterampilan yang dibutuhkan yang tidak dimiliki oleh staf yang ada. Cara lainnya adalah dengan mengalihdayakan proses ke organisasi yang memiliki keterampilan yang tidak dimiliki oleh staf yang ada. Kedua pendekatan ini kontroversial dan biasanya dipertimbangkan hanya ketika keterampilan baru yang dibutuhkan kemungkinan besar paling berbeda dari serangkaian keterampilan staf saat ini. Dalam kebanyakan kasus, organisasi memilih alternatif ketiga: melatih staf yang ada dalam proses bisnis baru dan sistem yang akan datang. Setiap rencana pelatihan harus mempertimbangkan apa yang akan dilatih dan bagaimana menyampaikan pelatihan.

Apa yang Harus Dilatih. Pelatihan apa yang harus Anda berikan kepada pengguna sistem? Sudah jelas: cara menggunakan sistem. Pelatihan harus mencakup semua kemampuan sistem baru sehingga pengguna memahami apa yang dilakukan setiap modul, bukan? Salah. Pelatihan untuk sistem bisnis harus berfokus pada membantu pengguna menyelesaikan pekerjaan mereka, bukan pada cara menggunakan sistem. Sistem hanyalah sarana untuk mencapai tujuan, bukan tujuan itu sendiri. Fokus pada melakukan pekerjaan (yaitu, proses bisnis), tidak menggunakan sistem, memiliki dua implikasi penting. Pertama, pelatihan harus fokus pada kegiatan di sekitar sistem serta pada sistem itu sendiri. Pelatihan harus membantu pengguna memahami bagaimana komputer cocok dengan gambaran yang lebih besar dari pekerjaan mereka. Penggunaan sistem harus diletakkan dalam konteks proses bisnis manual maupun yang terkomputerisasi, dan juga harus mencakup kebijakan manajemen baru yang diterapkan bersama dengan sistem komputer baru.

Kedua, pelatihan harus fokus pada apa yang perlu dilakukan pengguna, bukan apa yang dapat dilakukan sistem. Ini adalah perbedaan yang halus—tetapi sangat penting—. Sebagian besar sistem menyediakan kemampuan yang jauh lebih banyak daripada yang perlu digunakan pengguna (misalnya, kapan terakhir kali Anda menulis makro di Microsoft Word?). Daripada mencoba mengajari pengguna semua fitur sistem, pelatihan seharusnya berfokus pada rangkaian aktivitas yang jauh lebih kecil yang dilakukan pengguna secara teratur dan memastikan bahwa pengguna benar-benar ahli dalam hal itu. Ketika fokusnya adalah pada 20 persen fungsi yang akan digunakan pengguna 80 persen dari waktu (alih-alih mencoba untuk mencakup semua fungsi), pengguna menjadi yakin tentang kemampuan mereka untuk menggunakan sistem. Pelatihan harus menyebutkan fungsi-fungsi lain yang jarang digunakan tetapi hanya agar pengguna menyadari keberadaannya dan mengetahui bagaimana mempelajarinya ketika penggunaannya diperlukan.

Salah satu sumber panduan untuk merancang materi pelatihan adalah Use-Case. Use-Case menguraikan aktivitas umum yang dilakukan pengguna dan dengan demikian dapat membantu dalam memahami proses bisnis dan fungsi sistem yang mungkin paling penting bagi pengguna.

Cara Melatih. Ada banyak cara untuk memberikan pelatihan. Pendekatan yang paling umum digunakan adalah pelatihan kelas, di mana banyak pengguna dilatih pada saat yang sama oleh instruktur yang sama. Ini memiliki keuntungan melatih banyak pengguna pada satu waktu dengan hanya satu instruktur dan menciptakan pengalaman bersama di antara pengguna.

Dimungkinkan juga untuk memberikan pelatihan satu lawan satu, di mana satu pelatih bekerja sama dengan satu pengguna pada satu waktu. Ini jelas lebih mahal, tetapi pelatih dapat merancang program pelatihan untuk memenuhi kebutuhan pengguna individu dan dapat lebih memastikan bahwa pengguna benar-benar memahami materi. Pendekatan ini biasanya digunakan hanya ketika pengguna sangat penting atau ketika pengguna sangat sedikit.

Pendekatan lain yang menjadi lebih umum adalah dengan menggunakan beberapa bentuk pelatihan berbasis komputer/*computer-based training* (CBT), di mana program pelatihan disampaikan melalui komputer, baik dalam bentuk CD atau melalui Web. Program CBT dapat mencakup slide teks, audio, dan bahkan video dan animasi. CBT biasanya lebih mahal untuk dikembangkan tetapi lebih murah untuk disampaikan karena tidak diperlukan instruktur untuk benar-benar memberikan pelatihan.

Gambar 13-8 merangkum empat faktor penting untuk dipertimbangkan dalam memilih metode pelatihan: biaya pengembangan, biaya pengiriman, dampak, dan jangkauan. CBT biasanya lebih mahal untuk dikembangkan daripada pelatihan satu lawan satu atau kelas, tetapi lebih murah untuk disampaikan. Pelatihan satu lawan satu memiliki dampak paling besar pada pengguna karena dapat disesuaikan dengan kebutuhan, pengetahuan, dan kemampuan pengguna yang tepat, sedangkan CBT memiliki dampak paling kecil. Namun, CBT memiliki jangkauan terbesar—kemampuan untuk melatih sebagian besar pengguna melalui jarak terjauh dalam waktu singkat—karena lebih mudah didistribusikan daripada pelatihan di kelas dan satu-satu, hanya karena tidak diperlukan instruktur.

	One-on-One Training	Classroom Training	Computer-Based Training
Cost to develop	Low to Medium	Medium	High
Cost to deliver	High	Medium	Low
Impact	High	Medium to High	Low to Medium
Reach	Low	Medium	High

Gambar 13-8 Memilih Metode Pelatihan

Gambar 13-8 menunjukkan pola yang jelas untuk sebagian besar organisasi. Jika hanya ada beberapa pengguna untuk dilatih, pelatihan satu lawan satu adalah yang paling efektif. Jika ada banyak pengguna untuk dilatih, banyak organisasi beralih ke CBT. Kami percaya bahwa penggunaan CBT akan meningkat di masa depan. Cukup sering, organisasi besar menggunakan kombinasi ketiga metode tersebut. Terlepas dari pendekatan mana yang digunakan, penting untuk memberi pengguna satu set materi yang mudah diakses yang dapat dirujuk lama setelah pelatihan berakhir (biasanya panduan referensi cepat dan satu set manual, baik di atas kertas maupun elektronik. membentuk).

13.6 KEGIATAN PASCA PELAKSANAAN

Tujuan kegiatan pasca-implementasi adalah pelembagaan penggunaan sistem baru—yaitu, menjadikannya cara yang normal, diterima, dan rutin dalam menjalankan proses bisnis. Kegiatan pasca implementasi mencoba untuk membekukan kembali organisasi setelah transisi berhasil ke sistem baru. Meskipun pekerjaan tim proyek secara alami mereda setelah

implementasi, sponsor bisnis dan terkadang manajer proyek secara aktif terlibat dalam pembekuan ulang. Keduanya—dan, idealnya, banyak pemegang kepentingan lainnya—secara aktif mempromosikan sistem baru dan memantau adopsi dan penggunaannya. Mereka biasanya memberikan aliran informasi yang stabil tentang sistem dan mendorong pengguna untuk menghubungi mereka untuk mendiskusikan masalah.

Pada bagian ini, kami memeriksa tiga kegiatan utama pasca implementasi: dukungan sistem (memberikan bantuan dalam penggunaan sistem), pemeliharaan sistem (terus menyempurnakan dan meningkatkan sistem), dan penilaian proyek (menganalisis proyek untuk memahami kegiatan apa yang dilakukan dengan baik—dan harus diulang—dan kegiatan apa yang perlu diperbaiki dalam proyek mendatang).

Dukungan Sistem

Setelah tim proyek menginstal sistem dan melakukan aktivitas manajemen perubahan, sistem secara resmi diserahkan ke grup operasi. Kelompok ini bertanggung jawab untuk mengoperasikan sistem, sedangkan tim proyek bertanggung jawab untuk mengembangkan sistem. Anggota kelompok operasi biasanya terlibat erat dalam kegiatan instalasi karena merekalah yang harus memastikan bahwa sistem benar-benar berfungsi. Setelah sistem terinstal, tim proyek pergi tetapi grup operasi tetap ada.

- Waktu dan tanggal laporan
- Nama, alamat email, dan nomor telepon orang pendukung yang menerima laporan
- Nama, alamat email, dan nomor telepon orang yang melaporkan masalah
- Perangkat lunak dan/atau perangkat keras yang menyebabkan masalah
- Lokasi masalah
- Deskripsi masalah
- Tindakan yang diambil
- Disposisi (masalah diperbaiki atau diteruskan ke pemeliharaan sistem)

Gambar 13-9 Elemen Laporan Masalah

Memberikan dukungan sistem berarti membantu pengguna untuk menggunakan sistem. Biasanya, ini berarti memberikan jawaban atas pertanyaan dan membantu pengguna memahami cara melakukan fungsi tertentu; jenis dukungan ini dapat dianggap sebagai pelatihan berdasarkan permintaan.

Dukungan online adalah bentuk pelatihan sesuai permintaan yang paling umum. Ini termasuk dokumentasi dan layar bantuan yang dibangun ke dalam sistem, serta situs web terpisah yang menyediakan jawaban atas pertanyaan yang sering diajukan/*frequently asked questions* (FAQ), yang memungkinkan pengguna menemukan jawaban tanpa menghubungi seseorang. Jelas, tujuan dari sebagian besar dukungan sistem adalah untuk memberikan dukungan online yang cukup baik sehingga pengguna tidak perlu menghubungi seseorang, karena menyediakan dukungan online jauh lebih murah daripada menyediakan seseorang untuk menjawab pertanyaan.

Sebagian besar organisasi menyediakan meja bantuan yang menyediakan tempat bagi pengguna untuk berbicara dengan seseorang yang dapat menjawab pertanyaan (biasanya melalui telepon tetapi terkadang secara langsung). Help desk mendukung semua sistem, bukan hanya satu sistem tertentu, sehingga menerima panggilan tentang berbagai macam perangkat lunak dan perangkat keras. Help desk dioperasikan oleh staf pendukung level-1 yang memiliki keterampilan komputer yang sangat luas dan mampu menanggapi berbagai permintaan, mulai dari masalah jaringan dan masalah perangkat keras hingga masalah dengan perangkat lunak komersial dan masalah dengan perangkat lunak aplikasi bisnis yang dikembangkan di- rumah.

Tujuan dari sebagian besar meja bantuan adalah agar staf pendukung level-1 menyelesaikan 80 persen permintaan bantuan yang mereka terima pada panggilan pertama. Jika masalah tidak dapat diselesaikan oleh staf pendukung level 1 , laporan masalah (Gambar 13-9) diselesaikan (seringkali menggunakan sistem komputer khusus yang dirancang untuk melacak laporan masalah) dan diteruskan ke anggota staf pendukung level-2.

Anggota staf pendukung level-2 adalah orang-orang yang mengetahui sistem aplikasi dengan baik dan dapat memberikan saran ahli. Untuk sistem baru, mereka biasanya dipilih selama fase implementasi dan menjadi akrab dengan sistem yang sedang diuji. Terkadang anggota staf pendukung level-2 berpartisipasi dalam pelatihan selama proses manajemen perubahan untuk menjadi lebih berpengetahuan tentang sistem, proses bisnis baru, dan pengguna itu sendiri.

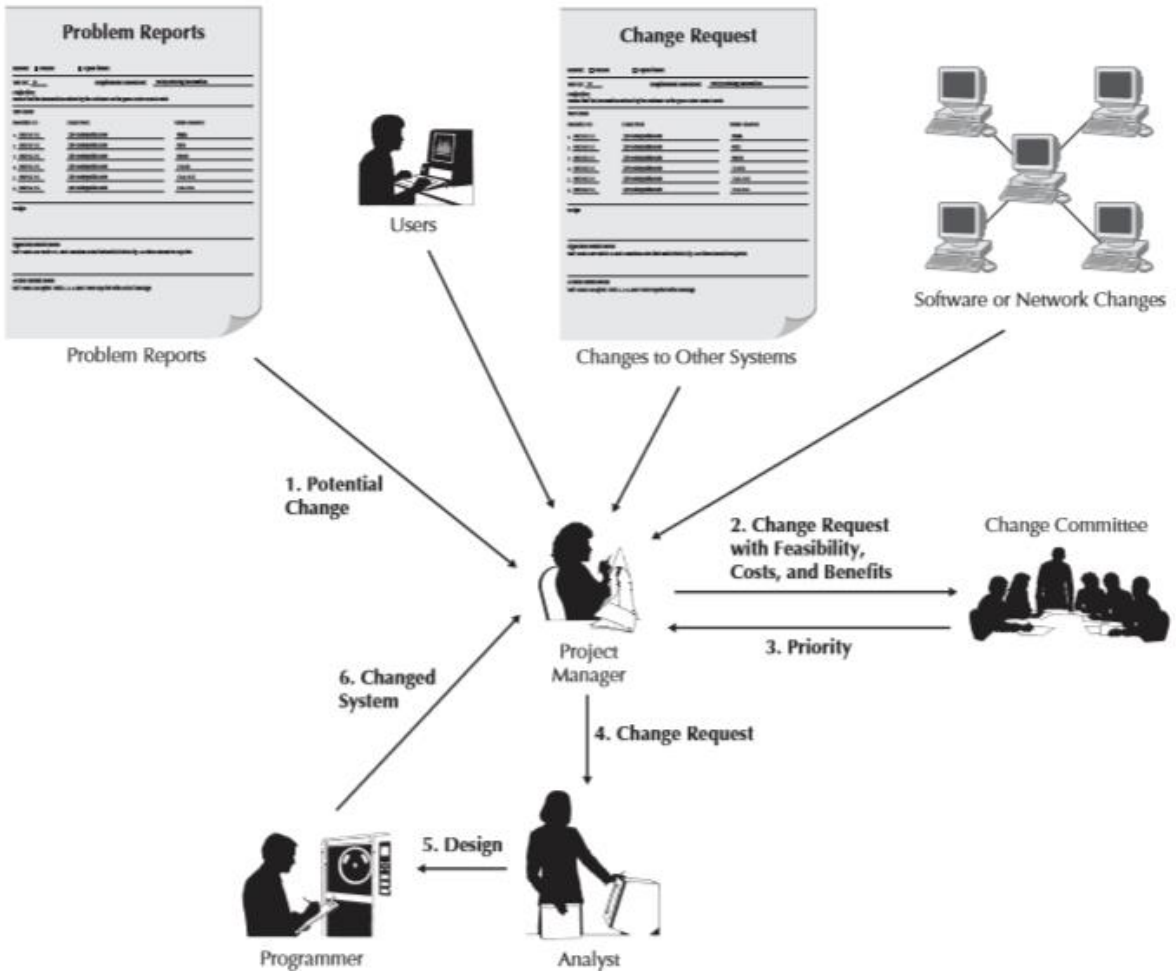
Staf pendukung level-2 bekerja dengan pengguna untuk menyelesaikan masalah. Sebagian besar masalah berhasil diselesaikan oleh staf level-2. Namun terkadang, terutama dalam beberapa bulan pertama setelah sistem diinstal, masalahnya ternyata adalah bug pada perangkat lunak yang harus diperbaiki. Dalam hal ini, laporan masalah menjadi permintaan perubahan yang diteruskan ke grup pemeliharaan sistem (lihat bagian berikutnya).

Perbaikan sistem

Pemeliharaan sistem adalah proses penyempurnaan sistem untuk memastikannya terus memenuhi kebutuhan bisnis. Lebih banyak uang dan usaha dicurahkan untuk pemeliharaan sistem daripada pengembangan awal sistem, hanya karena sistem terus berubah dan berkembang saat digunakan. Sebagian besar analis dan pemrogram sistem pemula bekerja terlebih dahulu pada proyek pemeliharaan; biasanya hanya setelah mereka memperoleh beberapa pengalaman mereka ditugaskan untuk proyek-proyek pembangunan baru.

Setiap sistem "dimiliki" oleh manajer proyek dalam grup IS (Gambar 13-10). Individu ini bertanggung jawab untuk mengoordinasikan upaya pemeliharaan sistem untuk sistem itu. Setiap kali perubahan potensial pada sistem diidentifikasi, permintaan perubahan disiapkan dan diteruskan ke manajer proyek. Permintaan perubahan adalah versi yang lebih kecil dari permintaan sistem yang dibahas dalam Bab 2. Ini menjelaskan perubahan yang diminta dan menjelaskan mengapa perubahan itu penting.

Perubahan bisa kecil atau besar. Permintaan perubahan yang mungkin memerlukan upaya yang signifikan biasanya ditangani dengan cara yang sama seperti permintaan sistem: Mereka mengikuti proses yang sama seperti proyek yang dijelaskan dalam buku ini, dimulai dengan identifikasi proyek di Bab 2 dan dilanjutkan hingga instalasi di bab ini. Perubahan kecil biasanya mengikuti versi yang lebih kecil dari proses yang sama ini. Ada penilaian awal kelayakan dan biaya dan manfaat, dan permintaan perubahan diprioritaskan. Kemudian analis sistem (atau programmer/analis) melakukan analisis, yang mungkin termasuk mewawancarai pengguna, dan menyiapkan desain awal sebelum pemrograman dimulai. Program baru (atau yang direvisi) kemudian diuji secara ekstensif sebelum sistem diubah dari sistem lama ke sistem yang direvisi.



Gambar 13-10 Memproses Permintaan Perubahan

Permintaan perubahan biasanya berasal dari lima sumber. Sumber yang paling umum adalah laporan masalah dari kelompok operasi yang mengidentifikasi bug dalam sistem yang harus diperbaiki. Ini biasanya diberikan prioritas langsung karena bug dapat menyebabkan masalah yang signifikan. Bahkan bug kecil dapat menyebabkan masalah besar dengan mengganggu pengguna dan mengurangi penerimaan dan kepercayaan mereka terhadap sistem.

Sumber permintaan perubahan kedua yang paling umum adalah peningkatan sistem dari pengguna. Saat pengguna bekerja dengan sistem, mereka sering mengidentifikasi perubahan kecil dalam desain yang dapat membuat sistem lebih mudah digunakan atau mengidentifikasi fungsi tambahan yang diperlukan. Peningkatan tersebut penting dalam memuaskan pengguna dan sering utama dalam memastikan bahwa perubahan sistem sebagai perubahan kebutuhan bisnis. Peningkatan sering diberikan prioritas kedua setelah perbaikan bug.

Sumber ketiga dari permintaan perubahan adalah proyek pengembangan sistem lainnya. Misalnya, jika dokter dalam masalah janji temu memutuskan bahwa dia ingin memiliki sistem janji temu berbasis Web yang memungkinkan pasien berinteraksi langsung dengan sistem janji temu saat ini, kemungkinan sistem lain, seperti penagihan, harus dimodifikasi untuk memastikan bahwa kedua sistem akan bekerja sama. Perubahan yang diperlukan oleh kebutuhan untuk mengintegrasikan dua sistem umumnya jarang terjadi tetapi menjadi lebih umum karena upaya integrasi sistem menjadi lebih umum.

Sumber permintaan perubahan keempat adalah yang terjadi ketika perangkat lunak atau jaringan yang mendasarinya berubah. Misalnya, versi baru Windows sering kali memerlukan aplikasi untuk mengubah cara sistem berinteraksi dengan Windows atau memungkinkan sistem aplikasi memanfaatkan fitur baru yang meningkatkan efisiensi. Meskipun pengguna mungkin tidak pernah melihat perubahan ini (karena sebagian besar perubahan ada di dalam sistem dan tidak memengaruhi antarmuka pengguna atau fungsionalitasnya), perubahan ini dapat menjadi salah satu yang paling menantang untuk diterapkan karena analis dan pemrogram harus mempelajari karakteristik sistem baru, memahami caranya sistem aplikasi menggunakan (atau dapat menggunakan) karakteristik tersebut, dan kemudian membuat perubahan pemrograman yang diperlukan.

Sumber kelima dari permintaan perubahan adalah manajemen senior. Permintaan perubahan ini sering didorong oleh perubahan besar dalam strategi atau operasi organisasi. Permintaan perubahan signifikan ini biasanya diperlakukan sebagai proyek terpisah, tetapi manajer proyek yang bertanggung jawab atas sistem awal sering kali ditugaskan untuk proyek baru.

Penilaian Proyek

Tujuan dari penilaian proyek adalah untuk memahami apa yang berhasil tentang sistem dan kegiatan proyek (dan, oleh karena itu, harus dilanjutkan dalam sistem atau proyek berikutnya) dan apa yang perlu ditingkatkan. Penilaian proyek tidak rutin di sebagian besar organisasi, kecuali untuk organisasi militer, yang terbiasa menyiapkan laporan setelah tindakan. Meskipun demikian, penilaian dapat menjadi komponen penting dalam pembelajaran organisasi karena membantu organisasi dan orang-orang memahami bagaimana meningkatkan pekerjaan mereka. Hal ini sangat penting bagi anggota staf junior karena membantu mempromosikan pembelajaran yang lebih cepat. Ada dua bagian utama dalam penilaian proyek—tinjauan tim proyek dan tinjauan sistem.

Tim proyek. Tinjauan Sebuah tinjauan tim proyek berfokus pada cara tim proyek melakukan kegiatannya. Setiap anggota proyek menyiapkan dokumen pendek dua sampai tiga halaman yang melaporkan dan menganalisis kinerjanya. Fokusnya adalah pada peningkatan kinerja, bukan hukuman atas kesalahan yang dilakukan. Dengan secara eksplisit mengidentifikasi kesalahan dan memahami penyebabnya, diharapkan anggota tim proyek akan lebih siap untuk menghadapi situasi serupa di lain waktu—dan cenderung tidak mengulangi kesalahan yang sama. Demikian juga, dengan mengidentifikasi kinerja yang sangat baik, anggota tim akan dapat memahami mengapa tindakan mereka berhasil dengan baik dan bagaimana mengulanginya di proyek mendatang.

Manajer proyek, yang bertemu dengan anggota tim untuk membantu mereka memahami bagaimana meningkatkan kinerja mereka, menilai dokumen yang disiapkan oleh setiap anggota tim. Manajer proyek kemudian menyiapkan dokumen ringkasan yang menguraikan pelajaran yang dipetik dari proyek. Ringkasan ini mengidentifikasi tindakan apa yang harus diambil dalam proyek masa depan untuk meningkatkan kinerja tetapi berhati-hati untuk tidak mengidentifikasi anggota tim yang membuat kesalahan. Ringkasan ini diedarkan secara luas di antara semua manajer proyek untuk membantu mereka memahami bagaimana mengelola proyek mereka dengan lebih baik. Seringkali, itu juga diedarkan di antara anggota staf biasa yang tidak mengerjakan proyek sehingga mereka juga dapat belajar dari proyek lain.

Tinjauan Sistem. Fokus dari tinjauan sistem adalah untuk memahami sejauh mana biaya dan manfaat yang diusulkan dari sistem baru yang diidentifikasi selama analisis kelayakan benar-benar diakui dari sistem yang diterapkan. Peninjauan tim proyek biasanya dilakukan segera setelah sistem diinstal saat peristiwa-peristiwa penting masih segar di benak anggota tim, tetapi tinjauan sistem sering dilakukan beberapa bulan setelah sistem diinstal

karena sering kali diperlukan beberapa saat sebelum sistem dapat berfungsi dengan baik. dinilai.

Tinjauan sistem dimulai dengan permintaan sistem dan analisis kelayakan yang disiapkan pada awal proyek. Analisis rinci disiapkan untuk nilai bisnis yang diharapkan (baik berwujud dan tidak berwujud) serta analisis kelayakan ekonomi diperiksa kembali, dan analisis baru disiapkan setelah sistem dipasang. Tujuannya adalah untuk membandingkan nilai bisnis yang diantisipasi dengan nilai bisnis aktual yang direalisasikan dari sistem. Ini membantu organisasi menilai apakah sistem benar-benar memberikan nilai yang direncanakan untuk diberikan. Apakah sistem memberikan nilai yang diharapkan atau tidak, proyek masa depan dapat memperoleh manfaat dari pemahaman yang lebih baik tentang biaya dan manfaat yang sebenarnya.

Tinjauan sistem formal juga memiliki implikasi perilaku penting untuk inisiasi proyek. Karena setiap orang yang terlibat dalam proyek mengetahui bahwa semua pernyataan tentang nilai bisnis dan perkiraan keuangan yang disiapkan selama inisiasi proyek akan dievaluasi pada akhir proyek, mereka memiliki insentif untuk bersikap konservatif dalam penilaian mereka. Tidak seorang pun ingin menjadi sponsor proyek atau manajer proyek untuk proyek yang secara radikal melampaui anggaran atau gagal memberikan manfaat yang dijanjikan.

Menghindari Buggy Software

Bagaimana Anda menghindari bug dalam perangkat lunak komersial yang Anda beli? Berikut enam tipsnya:

1. *Kenali perangkat lunak Anda*: Cari tahu apakah beberapa program yang Anda gunakan setiap hari memiliki bug dan tambalan yang diketahui, dan lacak situs web yang menawarkan informasi terbaru tentangnya.
2. *Cadangkan data Anda*: Diktum ini harus ditato di setiap monitor. Berhenti membaca sekarang dan salin data yang tidak dapat Anda hilangkan ke hard disk kedua atau server Web. Kami akan menunggu.
3. *Belum Upgrade*: Sangat menggoda untuk meningkatkan ke versi terbaru dan terbaik dari perangkat lunak favorit Anda, tetapi mengapa melakukannya? Tunggu beberapa bulan, periksa pengalaman pengguna lain dengan peningkatan di newsgroup Usenet atau forum diskusi vendor itu sendiri, lalu lakukan. Tetapi hanya jika Anda harus.
4. *Upgrade Perlahan*: Jika Anda memutuskan untuk meningkatkan, beri waktu setidaknya satu bulan untuk menguji pemutakhiran pada sistem terpisah sebelum Anda menginstalnya di semua komputer di rumah atau kantor Anda.
5. *Forget The Beta*: Menginstal perangkat lunak beta di komputer utama Anda adalah permainan roulette Rusia. Jika Anda benar-benar harus bermain dengan perangkat lunak beta, dapatkan komputer kedua.
6. *Komplain* : semakin Anda mengeluh tentang bug dan meminta perbaikan, semakin mahal bagi vendor untuk mengirimkan produk buggy. Ini seperti pemungutan suara—semakin banyak orang berpartisipasi, semakin baik hasilnya.

Berdasarkan materi dari "Software Bugs Run Rampant," PC World 17, no. 1 (Januari 1999): 46.

Pertanyaan

1. Apa tiga langkah dasar dalam mengelola perubahan organisasi?
2. Apa masalah budaya yang harus diperhatikan oleh pengembang?
3. Apa saja komponen utama dari rencana migrasi?
4. Bandingkan dan kontraskan konversi langsung dan konversi paralel.
5. Bandingkan dan kontraskan konversi percontohan, konversi bertahap, dan konversi simultan.
6. Bandingkan dan kontraskan konversi modular dan konversi seluruh sistem.
7. Jelaskan trade-off antara memilih antara jenis konversi dalam pertanyaan 4, 5, dan 6.
8. Apa tiga peran utama dalam inisiatif manajemen perubahan?
9. Mengapa orang menolak perubahan? Jelaskan model dasar untuk memahami mengapa orang menerima atau menolak perubahan.
10. Apa tiga elemen utama dari kebijakan manajemen yang harus dipertimbangkan ketika menerapkan sistem baru?
11. Bandingkan dan kontraskan strategi manajemen perubahan informasi dengan strategi manajemen perubahan politik. Apakah yang satu lebih baik dari yang lain?
12. Jelaskan tiga kategori pengadopsi yang mungkin Anda temui dalam inisiatif manajemen perubahan apa pun.
13. Bagaimana Anda harus memutuskan item apa yang akan disertakan dalam rencana pelatihan Anda?
14. Bandingkan dan kontraskan tiga pendekatan dasar untuk pelatihan.
15. Apa peran kelompok operasi dalam pengembangan sistem?
16. Bandingkan dan kontraskan dua cara utama dalam menyediakan dukungan sistem.
17. Bagaimana laporan masalah berbeda dari permintaan perubahan?
18. Apa sumber utama permintaan perubahan?
19. Mengapa penilaian proyek penting?
20. Bagaimana tinjauan tim proyek berbeda dari tinjauan sistem?
21. Menurut Anda apa tiga kesalahan umum yang dilakukan analis pemula dalam bermigrasi dari sistem saat in ke sistem yang akan datang?
22. Beberapa ahli berpendapat bahwa manajemen perubahan lebih penting daripada bagian lain dari pengembangan sistem. Apakah Anda setuju atau tidak? Menjelaskan.
23. Dalam pengalaman kami, perencanaan manajemen perubahan sering kali kurang mendapat perhatian daripada perencanaan konversi. Menurut Anda mengapa ini terjadi?

Latihan

A. Misalkan Anda menginstal paket akuntansi baru di bisnis kecil Anda. Strategi konversi apa yang akan Anda gunakan? Kembangkan rencana konversi (yaitu, hanya aspek teknis).
B. Misalkan Anda memasang sistem reservasi kamar baru untuk universitas Anda yang melacak kursus mana yang ditugaskan ke kamar mana. Asumsikan bahwa semua ruangan di setiap gedung "dimiliki" oleh satu perguruan tinggi atau departemen dan hanya satu orang di perguruan tinggi atau departemen tersebut yang memiliki izin untuk menugaskannya. Strategi konversi apa yang akan Anda gunakan? Kembangkan rencana konversi (yaitu, hanya aspek teknis).
C. Misalkan Anda memasang sistem penggajian baru di perusahaan multinasional yang sangat besar. Strategi konversi apa yang akan Anda gunakan? Kembangkan rencana konversi (yaitu, hanya aspek teknis).

D. Pertimbangkan perubahan besar yang Anda alami dalam hidup Anda (misalnya, mengambil pekerjaan baru, memulai sekolah baru). Siapkan analisis biaya-manfaat dari perubahan baik dari segi perubahan dan transisi ke perubahan.
E. Misalkan Anda adalah manajer proyek untuk sistem perpustakaan baru untuk universitas Anda. Sistem ini akan meningkatkan cara mahasiswa, fakultas, dan staf dapat mencari buku dengan memungkinkan mereka untuk mencari melalui Web, daripada hanya menggunakan sistem berbasis teks saat ini yang tersedia di terminal komputer di perpustakaan. Siapkan analisis biaya-manfaat dari perubahan baik dari segi perubahan dan transisi ke perubahan untuk pemegang kepentingan utama.
F. Menyiapkan rencana untuk memotivasi adopsi sistem dalam latihan E.
G. Siapkan rencana pelatihan yang mencakup apa yang akan Anda latih dan bagaimana pelatihan akan disampaikan untuk sistem dalam latihan
H. Misalkan Anda memimpin pemasangan DSS baru untuk membantu petugas penerimaan mengelola proses penerimaan di universitas Anda. Kembangkan rencana manajemen perubahan (yaitu, aspek organisasi saja).
I. Misalkan Anda adalah pemimpin proyek untuk pengembangan sistem pendaftaran kursus berbasis Web baru untuk universitas Anda yang menggantikan sistem lama di mana siswa harus pergi ke coliseum pada waktu-waktu tertentu dan mengantre untuk mendapatkan slip izin untuk setiap kursus mereka. ingin mengambil. Kembangkan rencana migrasi (termasuk konversi teknis dan manajemen perubahan).
J. Misalkan Anda adalah pemimpin proyek untuk pengembangan sistem reservasi maskapai baru yang akan digunakan oleh agen reservasi in-house maskapai. Sistem ini akan menggantikan sistem berbasis perintah saat ini yang dirancang pada tahun 1970-an yang menggunakan terminal. Sistem baru menggunakan PC dengan antarmuka berbasis Web. Kembangkan rencana migrasi (termasuk konversi dan manajemen perubahan) untuk operator telepon Anda.
K. Kembangkan rencana migrasi (termasuk konversi dan manajemen perubahan) untuk agen perjalanan independen yang menggunakan sistem reservasi maskapai yang dijelaskan dalam latihan J.
L. Untuk masalah A Real Estate Inc di Bab 4 sampai 12:
    a. Siapkan rencana untuk memotivasi adopsi sistem.
    b. Siapkan rencana pelatihan yang mencakup apa yang akan Anda latih dan bagaimana pelatihan akan disampaikan.
    c. Siapkan rencana manajemen perubahan.
    d. Mengembangkan rencana migrasi.
M. Untuk masalah A Video Store di Bab 4 sampai 12:
    a. Siapkan rencana untuk memotivasi adopsi sistem.
    b. Siapkan rencana pelatihan yang mencakup apa yang akan Anda latih dan bagaimana pelatihan akan disampaikan.
    c. Siapkan rencana manajemen perubahan.
    d. Mengembangkan rencana migrasi.
N. Untuk masalah gym di Bab 4 sampai 12:
    a. Siapkan rencana untuk memotivasi penerapan sistem.
    b. Siapkan rencana pelatihan yang mencakup apa yang akan Anda latih dan bagaimana pelatihan akan disampaikan.
    c. Siapkan rencana manajemen perubahan.
    d. Mengembangkan rencana migrasi.
O. Untuk masalah Piknik R Us di Bab 4 sampai 12
    a. Siapkan rencana untuk memotivasi penerapan sistem.
    b. Siapkan rencana pelatihan yang mencakup apa yang akan Anda latih dan bagaimana pelatihan akan disampaikan.
    c. Siapkan rencana manajemen perubahan.

d. Mengembangkan rencana migrasi.

P. Untuk masalah Klub Bulanan di Bab 4 hingga 12:
   a. Siapkan rencana untuk memotivasi penerapan sistem.
   b. Siapkan rencana pelatihan yang mencakup apa yang akan Anda latih dan bagaimana pelatihan akan disampaikan.
   c. Siapkan rencana manajemen perubahan.
   d. Mengembangkan rencana migrasi.

Minicase

1. Nancy adalah kepala departemen IS di MOTO Inc., sebuah perusahaan manajemen sumber daya manusia. Staf IS di MOTO Inc. menyelesaikan pekerjaan pada sistem perangkat lunak manajemen klien baru sekitar sebulan yang lalu. Nancy terkesan dengan kinerja stafnya dalam proyek ini karena firma tersebut sebelumnya tidak pernah mengerjakan proyek dengan skala sebesar ini secara internal. Salah satu tugas mingguan Nancy adalah mengevaluasi dan memprioritaskan permintaan perubahan yang masuk untuk berbagai aplikasi yang digunakan oleh perusahaan.

   Saat ini, Nancy memiliki lima permintaan perubahan untuk sistem klien di mejanya. Salah satu permintaan adalah dari pengguna sistem yang ingin beberapa perubahan format dibuat untuk laporan harian yang dihasilkan oleh sistem. Permintaan lain adalah dari pengguna yang ingin urutan opsi menu diubah pada salah satu menu sistem untuk lebih mencerminkan frekuensi penggunaan opsi tersebut. Permintaan ketiga datang dari departemen penagihan.

   Departemen ini melakukan penagihan melalui paket perangkat lunak penagihan. Pembaruan besar-besaran dari perangkat lunak ini sedang direncanakan, dan antarmuka antara sistem klien dan sistem tagihan perlu diubah untuk mengakomodasi struktur data perangkat lunak baru. Permintaan keempat tampaknya merupakan bug sistem yang terjadi setiap kali klien membatalkan kontrak (untungnya, kejadian yang jarang terjadi). Permintaan terakhir datang dari Susan, presiden perusahaan. Permintaan ini mengkonfirmasi rumor bahwa MOTO Inc. akan mengakuisisi bisnis baru lainnya. Bisnis baru mengkhususkan diri dalam penempatan sementara karyawan profesional dan ilmiah yang terampil dan mewakili area bisnis baru untuk MOTO Inc. Sistem perangkat lunak manajemen klien perlu dimodifikasi untuk menggabungkan pengaturan klien khusus yang terkait dengan perusahaan yang diakuisisi.

   Bagaimana Anda merekomendasikan agar Nancy memprioritaskan permintaan perubahan ini untuk sistem klien/manajemen?

2. Sky View Aerial Photography menawarkan berbagai layanan fotografi udara, video, dan pencitraan inframerah. Perusahaan telah berkembang dari hari-hari awal memotret rumah klien hingga statusnya saat ini sebagai spesialis gambar udara layanan lengkap. Sky View sekarang memiliki banyak kontrak dengan berbagai lembaga pemerintah untuk pekerjaan pemetaan dan survei udara. Sky View memiliki kantor di bandara, di mana ia menyimpan armada pesawat yang dilengkapi secara khusus. Sky View mengadakan kontrak dengan beberapa pilot lepas dan fotografer untuk beberapa pekerjaan di udara dan juga mempekerjakan beberapa pilot dan fotografer penuh waktu.

   Pemilik Sky View Aerial Photography baru-baru ini mengontrak perusahaan konsultan pengembangan sistem untuk mengembangkan sistem informasi baru untuk bisnis tersebut. Dengan bertambahnya jumlah kontrak, pesawat, penerbangan, pilot, dan fotografer, perusahaan mengalami kesulitan untuk mencatat secara akurat aktivitas

bisnisnya dan penggunaan armada pesawatnya . Sistem baru akan mengharuskan semua pilot dan fotografer untuk menggesek lencana ID melalui pembaca di awal dan akhir setiap penerbangan foto, bersama dengan informasi rekaman tentang pesawat yang digunakan dan klien yang dilayani pada penerbangan itu. Catatan-catatan ini harus dicocokkan dengan catatan penggunaan pesawat sebenarnya yang dipelihara dan dicatat oleh personel hanggar.

Staf kantor sangat menantikan pemasangan sistem baru. Sikap umum mereka adalah bahwa sistem akan mengurangi jumlah masalah dan kesalahan yang mereka temui dan akan membuat pekerjaan mereka lebih mudah. Pilot, fotografer, dan staf hanggar kurang antusias, karena tidak terbiasa memantau aktivitas mereka dengan cara ini.

a. Diskusikan faktor-faktor yang mungkin menghambat penerimaan sistem baru ini oleh pilot, fotografer, dan staf hanggar.
b. Diskusikan bagaimana strategi informasi dapat digunakan untuk memotivasi adopsi sistem baru di Sky View Aerial Photography.
c. Diskusikan bagaimana strategi politik dapat digunakan untuk memotivasi penerapan sistem baru di Sky View Aerial Photography.

3. Untuk masalah Kendaraan Perjalanan Liburan yang dijelaskan dalam Bab 5 sampai 12:
   a. Siapkan rencana untuk memotivasi adopsi sistem.
   b. Siapkan rencana pelatihan yang mencakup apa yang akan Anda latih dan bagaimana pelatihan akan disampaikan.
   c. Siapkan rencana manajemen perubahan.
   d. Kembangkan rencana migrasi.
4. Untuk masalah Manajemen Staf Profesional dan Ilmiah yang dijelaskan dalam Bab 4, dan 6 hingga 11:
   a. Siapkan rencana untuk memotivasi adopsi sistem.
   b. Siapkan rencana pelatihan yang mencakup apa yang akan Anda latih dan bagaimana pelatihan akan disampaikan.
   c. Siapkan rencana manajemen perubahan.
   d. Kembangkan rencana migrasi.

Referensi

A. Bahrami, Object-Oriented Systems Development using the Unified Modeling Language (New York: McGraw-Hill, 1999)

A.R. Dennis, J.S. Valacich, T. Connolly, and B.E. Wynne, "Process Structuring in Electronic Brainstorming," Information Systems Research 7, no. 2 (June 1996): 268–277.

Adam Greenfield, Everywhere: The Dawning If the Age of Ubiquitous Computing (Berkeley, CA: New Riders, 2006)

Adrian McEwen and Hakim Cassimally, Designing the Internet of Th ings (Chichester, West Sussex, UK: Wiley, 2014)

Akmal B. Chaudri and Roberto Zicari, Succeeding with Object Databases: A Practical Look at Today's Implementations with Java and XML (New York: Wiley, 2001)

Alan Dennis, Networking in the Internet Age (New York: Wiley, 2002

Alan Snyder, "Encapsulation and Inheritance in Object-Oriented Programming Languages," in N. Meyrowitz (ed.), OOPSLA '86 Conference Proceedings, ACM SigPlan Notices 21, no. 11 (November 1986)

Alan Wexelblat (ed.), Virtual Reality: Applications and Explorations (Boston, MA: Academic Press, 1993)

Ali Bahrami, Object-Oriented Systems Development Using the Unified Modeling Language (New York: McGraw-Hill, 1999)

Alton R. Kindred, Data Systems and Management: An Introduction to Systems Analysis and Design, 2nd Ed. (Englewood Cliffs, NJ: Prentice-Hall, 1980).

Andrew Nusca, "Smart Takes: HP Opens First Wind-Cooled Green Data Center; Most Efficient to Date," SMARTPLANET (February 11, 2010).

Ann Lantham Cudworth, Virtual World Design (Boca Raton, FL: CRC Press, 2014).

Anthony Giddons, The Constitution of Society: Outline of the Theory of Structure (Berkeley: University of California Press, 1984).

B. Hederson-Sellers and B. Unhelkar, OPEN modeling with UML (Harlow, England: Addison-Wesley, 2000).

B. Meyer, Object- Oriented Software Construction (Englewood Cliffs, NJ: Prentice Hall, 1994)

B. Schheiderman, Designing the User Interface: Strategies for Effective Human Computer Interaction, 3rd Ed. (Reading, MA: Addison-Wesley, 1998)

B. Shriver and P. Wegner (eds.), Research Directions in Object-Oriented Programming (Cambridge, MA: MIT Press, 1987).

Bellin and S. S. Simone, The CRC Card Book (Reading, MA: Addison-Wesley, 1997).

Ben Fry, Visualizing Data (Sebastopol, CA: O'Reilly Media, 2008)

Bernard Suits, The Grasshopper: Games, Life, and Utopia (Ontario, CA: Broadview Press, 2005).

Bernd Brugge and Allen H. Dutoit, Object-Oriented Software Engineering: Conquering Complex and Changing Systems (Englewood Cliffs, NJ: Prentice Hall, 2000). 10)

Bertrand Meyer, Object-Oriented Software Construction (Englewood Cliffs, NJ: Prentice Hall, 1988).

Bo Begole, Ubiquitous Computing for Business (Upper Saddle River, NJ: FT Press, 2011)

Brett C. Tjaden, Fundamentals of Secure Computer Systems (Wilsonville, OR: Franklin, Beedle, and Associates, 2004)

Bryon Reeves and J. Leighton Read, Total Engagement: Using Games and Virtual Worlds to Change the Way People Work and Businesses Compete (Boston, MA: Harvard Business Press, 2009).

Bryon Reeves and J. Leighton Read, Total Engagement: Using Games and Virtual Worlds to Change the Way People Work and Businesses Compete (Boston, MA: Harvard Business Press, 2009).

C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, and S. Angel, A Pattern Language (New York: Oxford University Press, 1977).

C. Larman, Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design (Englewood Cliffs, NJ: Prentice Hall, 1998).

Caril Baroudi, Jeff rey Hill, Arnold Reinhold, and Jhana Senxian, Green IT for Dummies™ (Hoboken, NJ: Wiley, 2009).

Chaomei Chen, Information Visualization and Virtual Environments (London: Springer-Verlag, 1999)

Chaomei Chen, Information Visualization and Virtual Environments (London: Springer-Verlag, 1999)

Chaomei Chen, Information Visualization: Beyond the Horizon, 2nd Ed. (London: Springer-Verlag, 2004)

Chris Stevens, Designing for the iPadTM: Building Applications that Sell (Chichester, UK: Wiley, 2011).

Craig Larman, Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design (Englewood Cliffs, NJ: Prentice Hall, 1998)

Dan Saffer, Designing Gestural Interfaces (Sebastopol, CA: O'Reilly, 2008).

Daniel P. Freedman and Gerald M. Weinberg, Handbook of Walkthrough, Inspections, and Technical Reviews: Evaluating Programs, Projects, and Products, 3rd Ed. (New York: Dorset House Publishing, 1990).

Daryl Conner, Managing at the Speed of Change (New York: Villard Books, 1992)

David C. Hay, Data Model Patterns: Conventions of Thought (New York: Dorset House, 1996)

David Rose, Enchanted Objects: Design, Human Desire, and the Internet of Th ings (New York, NY: Scribner, 2014)

David Tegarden, “Business Information Visualization,” Communications of the Association for Information Systems, Vol. 1, Article 4 (1999) (http://aisel.aisnet.org/cais/vol1/iss1/4).

Deborah Hix and H. Rex Hartson, Developing User Interfaces, Ensuring Usability Th rough Product & Process (New York: Wiley, 1993).

Dennis C. Brewer, Security Controls for Sarbanes–Oxley Section 404 IT Compliance: Authorization, Authentication, and Access (Indianapolis: Wiley, 2006).

Derek L. Hansen, Ben Shneiderman, and Marc A. Smith, Analyzing Social Media Networks with NodeXL: Insights from a Connected World (Burlington, MA: Morgan Kaufmann, 2011)

Dimitris Chorafas and Heinrich Steinmann, Virtual Reality: Practical Applications in Business and Industry (Englewood Cliffs, NJ: Prentice Hall, 1995)

Donn Felker, Android™ Application Development for Dummies™ (Hoboken, NJ: Wiley, 2011)

Dorothy E. Leidner and Timothy Kayworth, “A Review of Culture in Information Systems Research: Toward a Theory of Information Technology Culture Confl ict,” MIS Quarterly 30, no. 2 (2006): 357–399.

Douglas K. Barry, Web Services and Service-Oriented Architectures (San Francisco: Morgan Kaufman, 2003).

Douglas Smith, Taking Charge of Change (Reading, MA: Addison-Wesley, 1996)

Dusty Reagan, Twitter™ Application Development for Dummies™ (Hoboken, NJ: Wiley, 2010).

E. Carmel, Offshoring Information Technology: Sourcing and Outsourcing to a Global Workforce (Cambridge, England: Cambridge University Press, 2005)

E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software (Reading, MA: Addison-Wesley, 1995)

E. Yourdon, Modern Structured Analysis (Englewood Cliffs, NJ: Prentice Hall, 1989).

Edward R. Tufte, The Visual Display of Quantitative Information, Envisioning Information (Cheshire, CT: Graphics Press, 2001)

Edward T. Hall, Beyond Culture (New York: Anchor Books, 1981).

Elisa M. del Galdo and Jakob Nielsen, International User Interfaces (New York, NY: Wiley, 1996)

Ellis Horowitz and Sartaj Sahni, Fundamentals of Data Structures (Rockville, MD: Computer Science Press, 1982)

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, Design Patterns: Elements of Reusable Object Oriented Software (MA: Addison-Wesley, 1995).

Erik Brynjolfsson and Andrew McAfee, The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies (New York, NY: Norton, 2014).

Erin Malone, Designing Social Interfaces (Sebastopol, CA: O'Reilly, 2009) and Gavin Bell, Building Social Web Applications: Establishing Community at the Heart of Your Site (Sebastopol, CA: O'Reilly, 2009

F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, Pattern-Oriented Software Architecture: A System of Patterns (Chichester, UK: Wiley, 1996)

F. R. McFadden, J. A. Hoffer, and Mary B. Prescott, Modern Database Management, 4th Ed. (Reading, MA: Addison-Wesley, 1998)

Francis da Costa, Rethinking the Internet of Things: A Scalable Approach to Connecting Everything (New York, NY: Apress Media, 2013).

Gabe Zichermann and Joselin Linder, The Gamification Revolution: How Leaders Leverage Game Mechanics to Crush the Competition (New York: McGraw-Hill, 2013)

Geert Hofstede, Culture's Consequences: Comparing Values, Behaviors, Institutions and Organizations across Nations, 2nd Ed. (Thousand Oaks, CA: Sage, 2001)

Geert Hofstede, Gert Jan Hofstede, and Michael Minkov, Cultures and Organizations: Software of the Mind, 3rd Ed. (New York: McGraw-Hill, 2010)

Gershon Dublon and Joseph A. Paradiso, "Extra Sensory Perception," Scientific American 311, no. 1 (July 2014): 36–41.

Glenford J. Myers, Composite/Structured Design (New York, NY: Van Nostrand Reinhold, 1978).

Google Data Centers, Renewable Energy. Retrieved August 2014 from www.google.com/about/datacenters/renewable.

Grady Booch, James Rumbaugh, and Ivar Jacobson, The Unified Modeling Language User Guide (Reading, MA: Addison-Wesley, 1999).

Graham, Migrating to Object Technology (Wokingham, England: Addison-Wesley, 1995)

Gregory Kipper and Joseph Rampolla, Augmented Reality: An Emerging Technologies Guide to AR (Waltham, MA: Syngress, 2013).

H. Baetjer, Software as Capital: An Economic Perspective on Software Engineering (Los Alamitos, CA: IEEE Computer Society Press, 1997).

H.-E. Eriksson and M. Penker, Business Modeling with UML: Business Patterns at Work (New York: Wiley, 2000)

Howard Gardner, Frames of Mind: The Theory of Multiple Intelligences (New York: Basic Books, 1983).

I. Englander, The Architecture of Computer Hardware, Systems Software, & Networking: An Information Technology Approach (Hoboken, NJ: Wiley, 2009)

I. Graham, B. Henderson-Seller, and H. Yanoussi, The Open Process Specification (Reading, MA: Addison-Wesley, 1997)

I. Graham, Migrating to Object Technology [Reading, MA: Addison-Wesley, 1994)

I. Jacobson, G. Booch, and J. Rumbaugh, The Unified Software Development Process (Reading, MA: Addison-Wesley, 1999)

Ian Graham, Object-Oriented Methods: Principles & Practice, 3rd Ed. (Reading, MA: Addison-Wesley, 2001).

Irv Englander, The Architecture of Computer Hardware and Systems Software: An Information Technology Approach, 5th Ed. (Hoboken, NJ: Wiley, 2014)

J. Brachman, "I Lied about the Trees Or, Defaults and Definitions in Knowledge Representation," AI Magazine 5, no. 3 (Fall 1985): 80–93.

J. K. Halvey and B. M. Melby, Information Technology Outsourcing Transactions: Process, Strategies, and Contracts, 2nd Ed. (Hoboken, NJ: Wiley, 2005)

J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, Object-Oriented Modeling and Design (Englewood Cliffs, NJ: Prentice Hall, 1991).

J. Tidwell, Designing Interfaces: Patterns for Eff ective Interaction Design, 2nd Ed. (Sebastopol, CA: O'Reilly Media, 2010)

Jakob Nielsen and Robert Mack (eds.), Usability Inspection Methods (New York: Wiley, 1994). See also www.useit.com/papers.

James Paul Gee, Why Video Games Are Good for Your Soul (Champaign, IL: Common Ground Publishing, 2005)

James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen, Object-Oriented Modeling and Design (Englewood Cliffs, NJ: Prentice Hall, 1991)

Jane McGonigal, Reality Is Broken: Why Games Make Us Better and How They Can Change the World (New York: Penguin Books, 2011)

Jenifer Tidwell, Designing Interfaces: P atterns for Effective Design, 2nd Ed. (Sebastopol, CA: O'Reilly, 2010).

Jesse Schell, The Art of Game Design: A Book of Lenses (Boca Raton, FL: CRC Press, 2008).

Jesse Stay, Facebook™ Application Development for Dummies™ (Hoboken, NJ: Wiley, 2011)

John D. McGregor and David A. Sykes, A Practical Guide to Testing Object-Oriented Soft]ware (Boston: Addison-Wesley, 2001).

John Satzinger and Lorne Olfman, "User Interface Consistency Across End-User Application: The Effects of M ental Models," Journal of Management Information Systems (Spring 1998): 167–193.

John Yunker, Beyond Borders: Web Globalization Strategies (Berkley, CA: New Riders, 2003).

Jon Radoff , Game On: Energize Your Business with Social Media Games (Indianapolis, IN: Wiley, 2011)

Jos Warmer and Anneke Kleppe, The Object Constraint Language: Precise Modeling with UML (Reading, MA: Addison-Wesley, 1999).

Judith Hurwitz, Marcia Kaufman, Fern Halper, and Robin Bloor, Cloud Computing for Dummies™ (Hoboken, NJ: Wiley 2010).

Judith R. Brown, Rae Earnshaw, Mikael Jern, and John Vince, Visualization: Using Computer Graphics to Explore Data and Present Information (New York: Wiley, 1995)

K. Beck and W. Cunningham, "A Laboratory for Teaching Object-Oriented Th inking," Proceedings of OOPSLA, SIGPLAN Notices, 24, no. 10 (1989): 1–6

K.K. Hausman and S. Cook, IT Architecture for Dummies (Hoboken, NJ: Wiley Publishing, 2011).

Kalani Kirk Hausman and Susan L. Cook, IT Architecture for Dummies™ (Hoboken, NJ: Wiley, 2011).

Karl J. Lieberherr and Ian M. Holland, "Assuring Good Style for Object-Oriented Programs," IEEE Software 6, no. 5 (September, 1989): 38–48

Karl J. Lieberherr, Adaptive Object-Oriented Software: The Demeter Method with Propagation Patterns (Boston, MA: PWS Publishing, 1996).

Kent L. Norman, The Psychology of Menu Selection (Norwood NJ: Ablex Publishing Corp., 1991).

Kevin Werbach and Dan Hunter, For the Win: How Game Thinking Can Revolutionize Your Business (Philadelphia, AA: Wharton Digital Press, 2012).

Kris Duggin and Kate Shoup, Business Gamification for Dummies (Hoboken, NJ: Wiley, 2013).

Kurt Lewin, "Frontiers in Group Dynamics," Human Relations 1, no. 5 (1947): 5–41; Kurt Lewin, "Group Decision and Social Change," in E. E. Maccoby, T. M. Newcomb, and E. L. Hartley (eds.), Readings in Social Psychology (New York: Holt, Rinehart, & Winston, 1958), pp. 197–211.

L. Silverston, The Data Model Resource Book: A Library of Universal Data Models for All Enterprises, Volume 1, Revised Ed. (New York, NY; Wiley, 2001).

L. Willcocks and G. Fitzgerald, A Business Guide to Outsourcing Information Technology (London: Business Intelligence, 1994)

Lee Sheldon, The Multiplayer Classroom: Designing Coursework as a Game (Boston, MA: Course Technology, 2012)

M. Blaha and W. Premerlani, Object-Oriented Modeling and Design for Database Applications (Englewood Cliffs, NJ: P rentice Hall, 1998)

M. Fowler with K. Scott, UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd Ed. (Reading, MA: Addison-Wesley, 2004).

M. Fowler, Analysis Patterns: Reusable Object Models (Reading, MA: Addison-Wesley, 1997)

M. Gillenson, Fundamentals of Database Management Systems (Hoboken, NJ: John Wiley & Sons, 2005)

M. Lacity and R. Hirschheim, Information Systems Outsourcing: Myths, Metaphors, and Realities (New York, NY: Wiley, 1993)

M. Lenzerini, D. Nardi, and M. Simi, Inheritance Hierarchies in Knowledge Representation and Programming Languages (New York: Wiley, 1991).

M. Page-Jones, Fundamentals of Object-Oriented Design in UML (Reading, MA: AddisonWesley, 2000)

Mary E. S. Loomis, Data Management and File Structures, 2nd Ed. (Englewood Cliffs, NJ: Prentice Hall, 1989)

Mary Lynn Manns and Linda Rising, Fearless Change: Patterns for Introducing New Ideas (Boston: Addison-Wesley, 2005).

Meilir Page-Jones, "Comparing Techniques by Means of Encapsulation and Connascence," Communications of the ACM 35, no. 9 (September 1992): 147–151.

Meilir Page-Jones, Fundamentals of Object-Oriented Design in UML (Reading, MA: Addison-Wesley, 2000).

Meilir Page-Jones, Fundamentals of Object-Oriented Design in UML (New York: Dorset House, 2000)].

Meyer, Object-Oriented Software Construction; R. Wirfs-Brock, B. Wilkerson, and L. Wiener, Designing ObjectOriented Software (Englewood Cliffs, NJ: Prentice Hall, 1990).

Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Areil Rabkin, Ion Stoica, and Matei Zahara, "A View of Cloud Computing," Communications of the ACM 53, no. 4 (2010): 50–58.

Michael Blaha and William Premerlani, ObjectOriented Modeling and Design for Database Applications (Upper Saddle River, NJ: Prentice Hall, 1998)

Michael Fagan, "Design and Code Inspections to Reduce Errors in Program Development," IBM Systems Journal 15, no. 3 (1976)

Michael J. Folk and Bill Zoeellick, File Structures: A Conceptual Toolkit (Reading, MA: Addison-Wesley, 1987).

Michael T. Goodrich and Roberto Tamassia, Data Structures and Algorithms in Java (New York: Wiley, 1998).

N. Singh and A. Pereira, The Culturally Customized Web Site: Customizing Web Sites for the Global Marketplace (Oxford, UK: Elsevier, 2005).

Nathan Yau, Data Points: Visualization Th at Means Something (Indianapolis, IN: Wiley, 2013).

Nathan Yau, Visualize Th is: The FlowingData Guide to Design, Visualization, and Statistics (Indianapolis, IN: Wiley, 2011)

Neal Goldstein and Tony Bove, iPad™ Application Development for Dummies™ (Hoboken, NJ: Wiley, 2010)

Nitish Singh and Arun Pereira, The Culturally Customized Web Site: Customizing Web Sites for the Global Marketplace (Oxford, UK: Elsevier Butterworth Heinemann, 2005)

Owen Hanson, Design of Computer Data Files (Rockville, MD: Computer Science Press, 1982).

P. Coad and E. Yourdon, Object-Oriented Design (Englewood Cliffs, NJ: Yourdon Press, 1991)

P. Coad, D. North, and M. Mayfield, Object Models: Strategies, Patterns, & Applications, 2nd Ed. (Englewood Cliffs, NJ: Prentice Hall, 1997)

P. Ghandforoush, T.K. Sen, and D. Tegarden, R. Ramaswamy, "Designing Systems Using Business Components: A Case Study in Call Center Automation." International Journal of Electronic Customer Relationship Management 4, no. 2 (2010): 161–179.

P. Kruchten, The Rational Unified Process: An Introduction, 2nd ed. (Reading, MA: Addison-Wesley, 2000)

Patrick Connor and Linda Lake, Managing Organizational Change, 2nd Ed. (Westport, CT: Praeger, 1994)

Paul R. Read, Jr., Developing Applications with Java and UML (Boston: Addison-Wesley, 2002).

Pawel Plaszczak and Richard Welner, Jr., Grid Computing (San Francisco: Morgan Kaufman, 2006).

Peter Coad and Edward Yourdon, Object-Oriented Design (Englewood Cliffs, NJ: Yourdon Press, 1991), p. 128.

Pramod J. Sadalage and Martin Fowler, NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence (Upper Saddle River, NJ: Addison Wesley, 2013).

R. J. Muller, Database Design for Smarties: Using UML for Data Modeling (San Francisco: M organ Kaufmann, 1999).

R. Wirfs-Brock, B. Wilkerson, and L. Weiner, Designing Object-Oriented Software (Englewood Cliffs, NJ: Prentice Hall, 1990)

Rajat Paharia, Loyalty 3.0: How Big Data and Gamification Are Revolutionizing Customer and Employee Engagement (New York: McGraw-Hill, 2013)

Ralph Koster, A Theory of Fun for Game Design, 2nd Ed. (Sebastopol, CA: O'Reilly Media, 2014)

Reginald Golledge (ed.), Wayfinding Behavior: Cognitive Mapping and Other Spatial Processes (Baltimore, MD: The John Hopkins University Press, 1999)

Richard A. Posner, The Little Book of Plagiarism (New York: Pantheon Books, 2007).

Richarde E. Nisbett, The Geography of Thought: How Asians and Westerners Think Differently … And Why (New York: Free Press, 2003).

Rob Kitchin and Scott Freundschuh, Cognitive Mapping: Past, Present and Future (London: Routledge, 2000).

Robert Spence, Information Visualization (Harlow England: ACM Press, 2001)

Robert Thierauf, Virtual Reality Systems for Business (Westport, CN: Quorum Books, 1995).

Robert V. Binder, Testing Object-Oriented Systems: Models, Patterns, and Tools (Reading, MA: Addison-Wesley, 1999)

Roger C. Schank, Making Minds Less Well Educated than Our Own (Mahwah, NJ: Lawrence Erlbaum Associates, 2004).

S. D. Burd, Systems Architecture, 6th Ed. (Boston: Course Technology, 2011)

S. Krug, Don't Make Me Think: A Common Sense Approach to Web Usability (Berkeley, CA: New Riders Publishing, 2006)

S. Lewis, The Art and Science of Smalltalk: An Introduction to Object-Oriented Programming Using Visual-Works (Englewood Cliffs, NJ: Prentice Hall, 1995).

S. Shlaer and S. J. Mellor, Object-Oriented Systems Analysis: Modeling the World in Data (Englewood Cliffs, NJ: Yourdon Press, 1988).

S.W. Ambler, The Elements of UML 2.0 Style (Cambridge, England: Cambridge University Press, 2005).

Sourceforge, sourceforge.net/projects/cosmos/.

Steve Krug, Don't Make Me Think: A Common Sense Approach to Web Usability, 2nd Ed. (Berkeley, CA: 2nd Ed. New R iders, 2006).

T. L. Friedman, The World Is Flat: A Brief History of the Twenty-First Century, Updated and Expanded Edition (New York: Farrar, Straus, and Giroux, 2006).

Thomas T. Barker, Writing Software Documentation (Boston: Allyn & Bacon, 1998).

Tony Mullen, Prototyping Augmented Reality (Indianapolis, IN: Syngress, 2011).

Usama Fayyad, Georges Grinstein, and Andreas Wierse (eds.), Information Visualization in Data Mining and Knowledge Discovery (San Francisco: Morgan Kaufmann, 2002).

Wanda Orlikowski and Dan Robey, "Information Technology and the Structuring of Organizations," Information Systems Research 2, no. 2 (1991): 143–169.

Warmer and Kleppe, The Object Constraint Language: Precise Modeling with UML.

William Strunk and E. B. White, Elements of Style, 4th Ed. (Needham Heights, MA: Allyn & Bacon, 2000).

www.smartplanet.com/blog/smart-takes/hp-opens-firstwind-cooled-green-data-center-most-efficient-to-date.

www.webtrust.org

DESAIN & ANALISIS

Sistem Berorientasi Obyek dengan UML

Dr. Joseph Teguh Santoso, S.Kom., M.Kom.
Migunani, M.Kom.

PROFILE PENULIS 1

Dr. Joseph Teguh Santoso, S.Kom., M.Kom. adalah Rektor dari Univeristas Sains & Teknologi Komputer (Universitas STEKOM) Semarang yang memiliki banyak pengalaman praktis dalam bidang *E-Commerce* sejak tahun 2002. Beliau mempunyai 3 (tiga) toko *Official Online Store* di China untuk merek sepeda Raleigh, dengan omzet tahunan pada tahun 2019 mencapai lebih dari Rp. 35 Milyar Rupiah dan terus meningkat.
Dr. Joseph T.S memiliki lisensi tunggal sepeda merek "Raleigh" untuk penjualanOnline di seluruh China. di samping itu beliau juga memiliki pabrik sepeda dan sepeda listrik merek "Fengjiu", yaitu Pabrik Sepeda listrik yang masih tergolong kecil di China. Pengalaman beliau malang melintang di dunia *online* store di China seperti Alibaba, Tmall, Taobao, JD, Aliexpress sangat membantu mahasiswa untuk memiliki pengalaman teknis dan praktis untuk membuka toko *online* bersama beliau.

PROFILE PENULIS 2

Migunani M.Kom adalah Ketua Program Studi Sistem Informasi Universitas Sains dan Teknologi Komputer (Universitas STEKOM). Seorang pengajar, peneliti dan pengabdi dibidang ilmu komputasi khususnya di bidang sistem informasi. Sebagai narasumber beberapa seminar dan webinar tentang digital marketing dan investasi dan penambangan (mining) cryptocurrency juga sebagai praktisi. Menerima Penghargaan The Best Of Fifth Paper Award pada konferensi International Conference On Information System For Business Competitifness dan penerima penghargaan penyaji terbaik seminar hasil program peningkatan kapasitas riset (penelitian dosen muda) dari Direktur Riset dan Pengabdian Masyarakat Dirjen DIKTI. Pengajaran untuk matakuliah rekayasa perangkat lunak, datamining, analisa dan perancangan sistem, sistem penunjang keputusan, teknologi dan sistem informasi. Beberapa bidang penelitian yang dilakukan diantaranya dengan tema Successful Multimedia Learning For Vocational Schools Model Development, Sistem Manajemen Konten Untuk Perdagangan Elektronik (E-Commerce) Pada UKM Sebagai Upaya Optimalisasi Manajemen Produk, Transaksi Dan Pelanggan. Aset Digital Elearning Berbasis Scorm Sebagai Upaya Resource Sharing Multi Platform LMS Untuk Sekolah Menengah Kejuruan RPL.

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
JL. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-623-6141-73-1 (PDF)
9 786236 141731