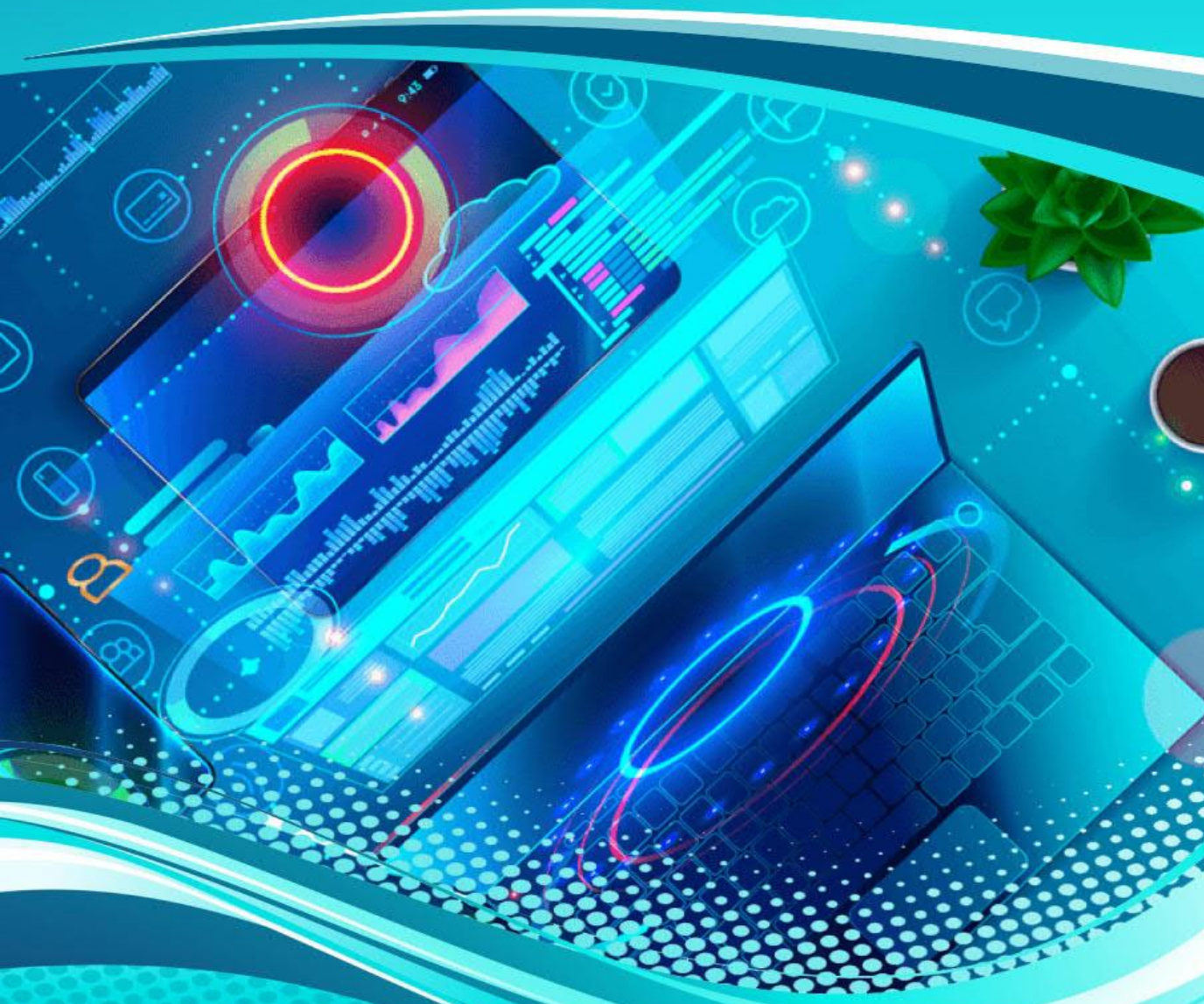
REKAYASA PERANGKAT LUNAK

MODEL PROSES MSF

Microsoft Solution Framework



Migunani, M.Kom

YAYASAN PRIMA AGUS TEKNIK

REKAYASA PERANGKAT LUNAK
MODEL PROSES MSF
Microsoft Solution Framework

Migunani. M,Kom

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 978-623-6141-26-7 (PDF)

Rekayasa Perangkat Lunak Model Proses MSF

Penulis:
Migunani, M.Kom

ISBN :

Editor:
Edy Siswanto., M.Kom

Penyunting :
Eko Siswanto, M.Kom

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

Penerbit :
Yayasan Prima Agus Teknik
Redaksi:
Jln Majapahit No 605 Semarang
Tlpn. (024) 6723456
Fax . 024-6710144
Email: penerbit_ypat@stekom.ac.id

Distributor Tunggal:
UNIVERSITAS STEKOM
Jln Majapahit No 605 Semarang
Tlpn. (024) 6723456
Fax . 024-6710144
Email: info@stekom.ac.id

KATA PENGANTAR

Assalamualikum Wr.Wb

Puji dan syukur penulis panjatkan kehadirat Allah SWT yang telah memberikan kekuatan dan kemampuan kepada penyusun untuk menyelesaikan buku kuliah ini. Buku kuliah ini diperuntukan bagi mahasiswa yang mengambil mata kuliah rekayasa perangkat lunak lanjutan menggunakan metodologi dari vendor Microsoft. Penulis berharap buku ajar ini dapat membantu mahasiswa dalam menguasai metode perancangan perangkat lunak menggunakan model proses Microsoft Solution Framework. Setelah membaca buku ini diharapkan mahasiswa dapat memahami mode; proses pengembangan perangkat lunak menggunakan alternatif yang umumnya digunakan adalah menggunakan system development lifecyce, model proses air terjun, prototyping.

Buku ini berisi pengantar perancangan dan model proses perangkat lunak, manajemen proyek model microsoft solution framework, mengumpulkan dan menganalisa informasi, tahap envisioning, tahap planning, mendesain lapisan presentasi, mendesain lapisan data, mendesain keamanan perangkat lunak, melengkapi tahapan planning dan tahap penstabilan dan penyebaran solusi.
Kritik dan saran yang membangun dari semua pihak terkait sangat penulis harapkan demi penyempurnaan modul perkuliahan ini.

Wassalamualaikum Wr.Wb

Penulis

Daftar Isi

BAGIAN I

PENGANTAR MODEL PROSES PERANCANGAN PERANGKAT LUNAK

1.1. Pendahuluan

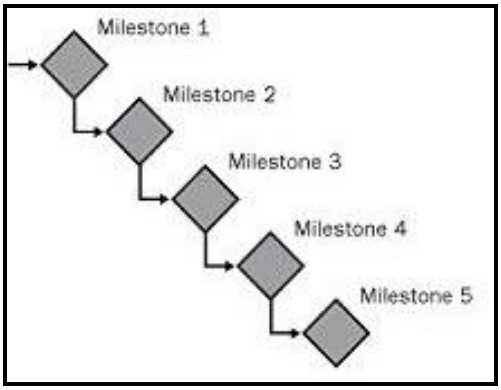
Dalam pengembangan sebuah perangkat lunak dibutuhkan suatu metodologi yang akan memandu tim pengembang dalam merancang perangkat lunak. Perancangan dimulai dengan menetapkan kebutuhan perangkat lunak tersebut sampai pada implementasi dan penyebaran atau distribusi.

Prinsip-prinsip perancangan perangkat lunak menjadi basis proses rekayasa aplikasi bisnis telah banyak di terapkan. Dimana sebuah metodologi merupakan kerangka pijakan utama dalam perancangan perangkat lunak profesional untuk menghasilkan aplikasi yang sesuai dengan kebutuhan bisnis sebuah organisasi. Tahapan-tahapan dalam membangun sebuah perangkat lunak akan sangat berguna untuk memastikan apakah elemen-elemen proyek pengembangan perangkat lunak telah dikelola dengan baik dan benar.

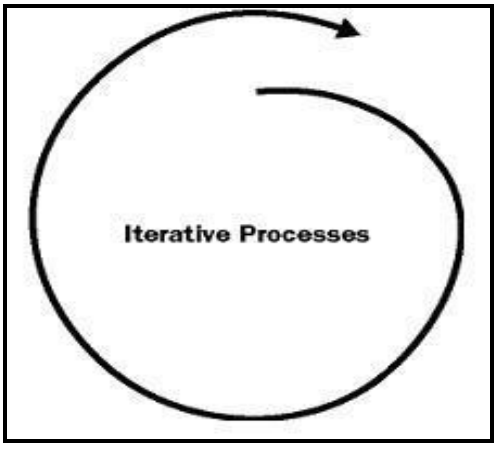
Microsoft Solution Framework (MSF) merupakan metodologi alternatif perancangan perangkat lunak yang menggabungkan dua metodologi yang berbeda menjadi satu kesatuan yang utuh untuk menghasilkan sebuah solusi perangkat lunak yang lebih dinamis dengan mengadopsi kelebihan dari masing-masing metodologi. Kedua metodologi tersebut adalah Waterfall dan Spiral. Metodologi Waterfall menggambarkan sebuah model proses yang statis dengan

tahapan-tahapan berlapis yang menggunakan sebuah milestone sebagai transisi pada setiap tahap perancangan, sementara metodologi Spiral (iterative) menerapkan model proses secara sirkular tanpa adanya cekpoint atau milestone. Namun kelebihan metodologi spiral adalah mengenai kebutuhan pengembangan secara keberlanjutan dan adanya keterlibatan pemakai dalam pembangunan perangkat lunak sehingga perangkat lunak akan selalu berkembang dengan versi dan fitur yang lebih baru.

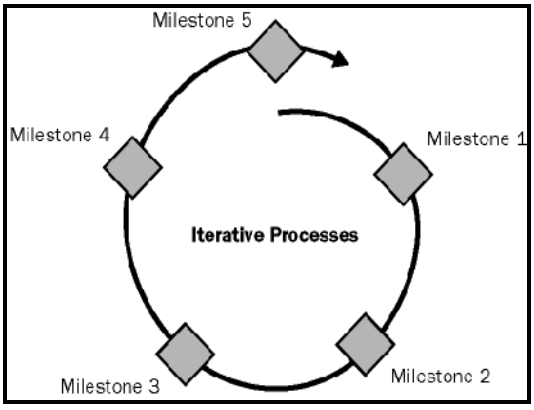
Keberhasilan pengembangan perangkat lunak bergantung pada pengelolaan proyek perangkat lunak secara keseluruhan. Menetapkan sebuah metodologi yang memiliki dinamisasi tinggi dalam tahap-tahap perancangan model proses perangkat lunak akan sangat berpengaruh pada kualitas perangkat lunak yang dihasilkan. Model proses merupakan tahap-tahap aktivitas proyek yang menggambarkan daur hidup suatu proyek. *Microsoft Solution Framework (MSF)* adalah metodologi perancangan dan pengembangan aplikasi bisnis yang diperkenalkan oleh sebuah vendor *software* besar yaitu *Microsoft Corporation*. Prinsip-prinsip pengembangan perangkat lunak dengan metodologi *MSF* memiliki perencanaan berbasis milestone (model waterfall), dan memberikan hasil yang dapat diprediksi (model spiral/iterative) disertai umpan balik dan kreativitas dari tim pengembang.



Gambar 1. Model Waterfall



Gambar 2. Model Spiral/Iterative



Gambar 3. Model Microsoft Solution Framework (MSF)

1.2. Model Proses Pengembangan Perangkat Lunak

Terdapat berbagai macam teknik pengembangan perangkat lunak seperti model waterfall, model spiral (iterative) dan model dari *Microsoft Solution Framework* atau yang lebih dikenal dengan *MSF, Model* yang dikembangkan oleh vendor besar teknologi informasi yaitu *Microsoft.*

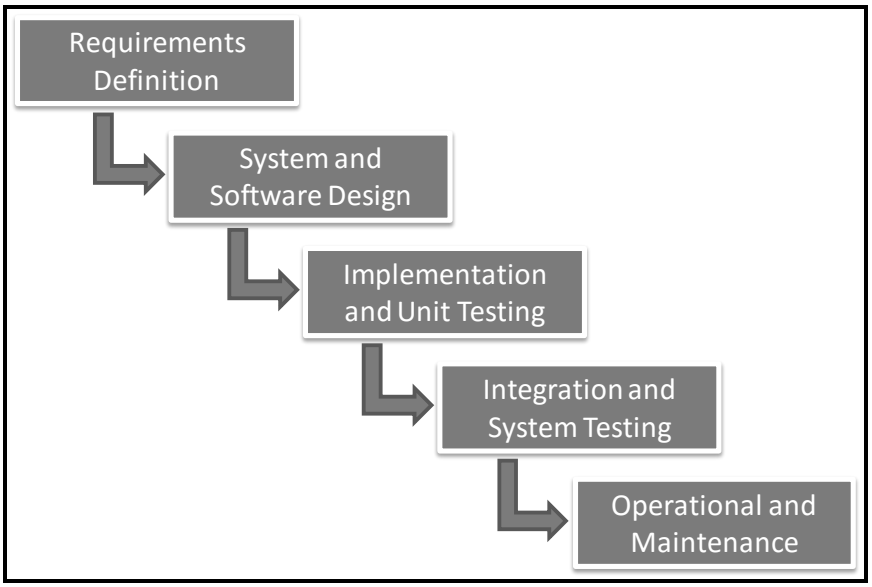
1.2.1. Model Proses Waterfall (*Sequential*)

Model ini menggunakan milestone sebagai titik transisi dan pengujian, artinya setiap aktivitas pada tahap pengembangan harus diselesaikan sebelum menuju tahap pengembangan berikutnya. Sehingga model ini sangat sesuai untuk perangkat lunak dengan syarat-syarat yang telah didefinisikan secara lengkap sebelumnya karena besar kemungkinan tidak adanya perubahan aplikasi dimasa yang akan datang. Kondisi semacam ini akan sangat berpengaruh pada perangkat lunak dan menimbulkan masalah terhadap kebutuhan iterasi dimana aplikasi akan terus berkembang dengan penyesuaian-penyesuaian terhadap kebutuhan, proses bisnis dan lingkungan aplikasi yang terus berubah dari waktu kewaktu. Namun kelebihan dari model ini adalah karena adanya titik transisi yang jelas pada setiap tahap, maka akan memudahkan tim pengembang perangkat lunak dalam memonitor penjadwalan proyek, menetapkan tanggung jawab dan akuntabilitas peran personal dalam proyek perangkat lunak.

Kelemahan dari model ini adalah adanya kendala dalam mengakomodasi perubahan setelah proses pengembangan telah berjalan. Fase sebelumnya harus lengkap dan selesai sebelum

memasuki tahap berikutnya. Beberapa kendala yang muncul pada model waterfall adalah :

a. Aspek perubahan pada perangkat lunak sulit dilakukan karena sifatnya yang kaku dimana kebutuhan perangkat lunak harus lengkap.
b. Karena sifat kakunya, model ini cocok manakala kebutuhan telah dikumpulkan secara lengkap sehingga perubahan bisa ditekan sekecil mungkin. ada kenyataannya sangat jarang sekali pengguna dapat mendefinisikan kebutuhan awal secara lengkap, oleh karena perubahan kebutuhan adalah sesuatu yang wajar terjadi.
c. Waterfall pada umumnya digunakan untuk rekayasa sistem perangkat lunak berkapasitas besar dimana proyek dikerjakan di beberapa tempat berlainan, dan terbagi menjadi beberapa bagian sub-proyek.



Gambar 4. Tahapan Pada Model Proses Waterfall

Sesuai dengan namanya "*waterfall*" atau air terjun, model *waterfall* menggunakan tahapan-tahapan seperti air terjun, dimana tahapan-tahapan tersebut terbagi menjadi lima tahapan.

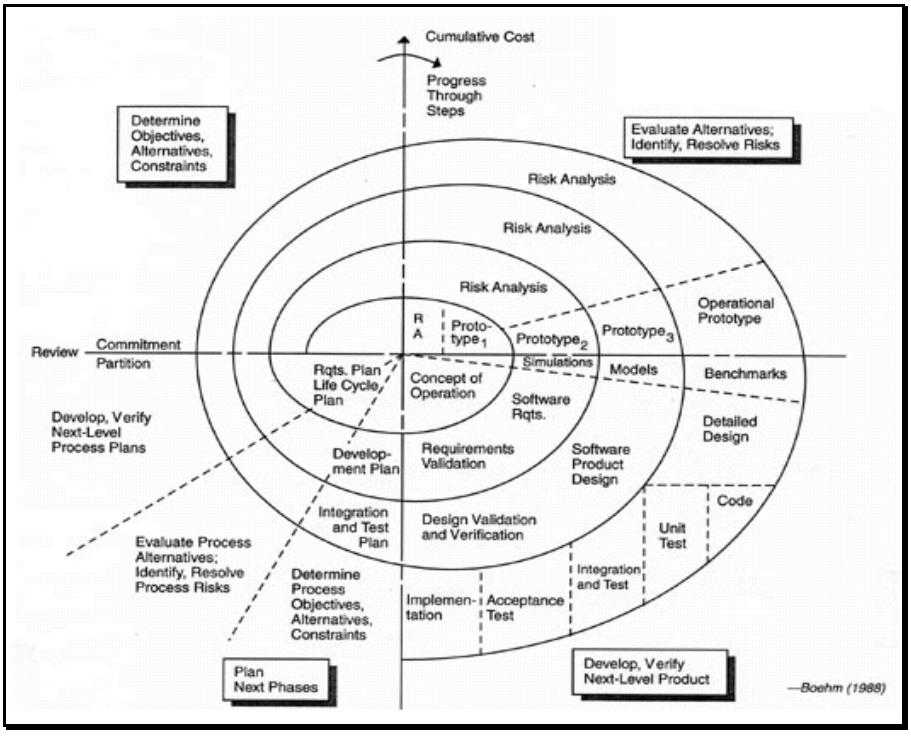
1. *Requirements analysis and definition* : pada tahap ini tim pengembang mengumpulkan kebutuhan secara lengkap kemudian dianalisis dan mendefinisikan kebutuhan-kebutuhan proses bisnis yang harus dipenuhi oleh perangkat lunak (solusi bisnis) yang akan dibangun.
2. *System and software design* : pada tahap ini tim pengembang mendesain sistem dan aplikasi meliputi desain konseptual, logikal dan fisikal.
3. *Implementation and unit testing* : pada tahap ini desain yang telah di rancang diimplementasikan dengan menterjemahkan ke dalam kode-kode program menggunakan sebuah bahasa pemrograman, sekaligus melakukan pengujian terhadap unit-unit program yang telah dibuat.
4. *Integration and system testing* : tim pengembang menyatukan unit-unit program kemudian melakukan pengujian sistem perangkat lunak secara keseluruhan.
5. *Operation and maintenance* : tim pengembang melakukan pengoperasian program dan melakukan pemeliharaan terhadap perangkat lunak dengan penyesuaian atau perubahan terhadap situasi sebenarnya.

1.2.2. Model Proses Spiral (*Iterative*)

Model ini berbasiskan pada kebutuhan terhadap aplikasi secara keberlanjutan untuk menyaring kebutuhan-kebutuhan

tersebut dan estimasi proyek secara keseluruhan. Model ini menerapkan perancangan model proses yang lebih dinamis dengan terus beradaptasi terhadap kebutuhan proses bisnis dimasa yang akan datang sehingga versi aplikasi terus berkembang dengan fitur-fitur yang mengalami peningkatan dari waktu kewaktu.

Kebutuhan waktu untuk pengembangan aplikasi yang cepat dengan kapasitas proyek yang relatif kecil sangat relefan dengan model spiral ini. Keterlibatan pelanggan dengan tim pengembang perangkat lunak akan sangat sering terjadi karena pelanggan akan memberikan *feedback* dan persetujuan setiap tahap dalam pengembangan aplikasi perangkat lunak. Dengan adanya *feedback* dari pelanggan maka estimasi waktu terhadap penyelesaian proyek perangkat lunak menjadi semakin jelas.



Gambar 5. Tahapan Pada Model Proses Spiral

Pengembangan perangkat lunak dengan model spiral memiliki kelemahan karena tidak adanya *milestone* sebagai titik transisi dan pengujian maka dikhawatirkan proses pengembangan sistem akan mengalami kekacauan dari segi waktu penyelesaian solusi sistem. Oleh karenanya model ini hanya sesuai untuk aplikasi-aplikasi kecil yang tidak terintegrasi dan terdistribusi.

Model spiral terbagi menjadi empat *quadrant,* dimana setiap *quadrant* merepresentasikan sebuah manajemen proses dengan tahapan-tahapan *identify, design, construct* dan *evaluate* (Dean Muench, 1994). Sistem akan melalui tahapan-tahapan proses yang akan berulang sebagai berikut :

1. Mendefinisikan tujuan dan kebutuhan bisnis, mengembangkan desain konseptual, rancangan konsep, rencana pengujian, dan analisis terhadap resiko dengan melibatkan pemakai.
2. Mendefinisikan kebutuhan sistem, mengembangkan desain logikal, mengkompilasi (*software-build*) rancangan awal, mengevaluasi hasil dengan melibatkan pemakai.
3. Mendifinisakan kebutuhan subsistem, menghasilkan desain fisikal, mengkompilasi rancangan berikutnya, mengevaluasi hasil dengan melibatkan pemakai.
4. Mendefinisikan kebutuhan setiap unit, menghasilkan desain akhir, mengkompilasi rancangan akhir, mengevaluasi keseluruhan level sistem dan penerimaan dari pemakai.

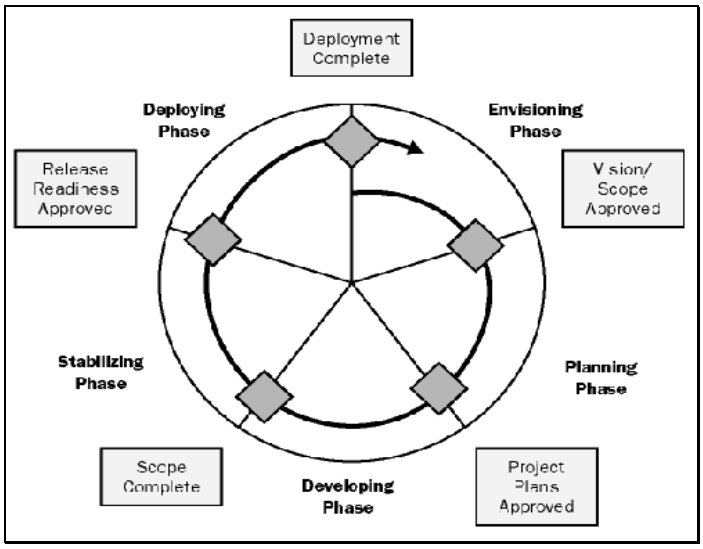
1.2.3. Model Proses Microsoft Solution Framework (MSF)

Model ini menggabungkan dua macam model proses terdahulu dengan menerapkan prinsip-prinsip terbaik dari masing-

masing model yaitu pada model *waterfall* dimana titik transisi yang jelas pada setiap tahap pengembangan sistem, maka akan memudahkan tim pengembang perangkat lunak dalam memonitor penjadwalan proyek, menetapkan tanggung jawab dan akuntabilitas peran personal dalam proyek perangkat lunak, sedangkan pada model spiral berbasiskan pada kebutuhan keberlanjutan untuk menyaring kebutuhan-kebutuhan sistem dan estimasi proyek secara keseluruhan. Selain itu model ini menerapkan perancangan model proses yang lebih dinamis dengan terus beradaptasi terhadap kebutuhan proses bisnis dimasa yang akan datang sehingga aplikasi memiliki versi yang berbeda dengan fitur-fitur yang mengalami peningkatan dari waktu kewaktu.

Model proses *Microsoft Solution Framework* akan menguraikan urutan aktivitas untuk membangun dan menyebarkan solusi perusahaan (*enterprise*), fleksibel dan dapat mengakomodasi pengembangan dan desain yang luas, berbasiskan fase (tahapan) seperti pada model waterfall, dan diterapkan untuk pengembangan perangkat lunak dengan *versioning*.

Model proses MSF memiliki tahapan-tahapan yang berbeda dengan model proses waterfall dan spiral. MSF mengenal lima tahapan yang disebut dengan *Envisioning Phase, Planning Phase, Developing Phase, Stabilizing Phase,* dan *Deploying Phase*. Pada setiap phase merupakan tahapan-tahapan yang berbeda dan mencapai kulminasi pada sebuah *milestone* dan mengasilkan *derivelable*.



Gambar 6. Tahapan Pada Model Proses MSF

1. *Envisioning Phase*.

Tim pengembang menguraikan secara lengkap mengenai tujuan dan batasan proyek, mengidentifikasi anggota tim dan apa yang harus dipenuhi oleh tim proyek kepada pemakai. Tujuan pada fase ini adalah membangun visi bersama proyek perangkat lunak yang melibatkan semua *stake holder*.

Milestone pada fase *envisioning* yaitu :

a) Tim inti proyek perangkat lunak telah di diorganisasikan.

b) Ruang lingkup proyek telah didefinisikan.

Deliverables (hasil yang diberikan) pada fase *envisioning* yaitu :

1. Ruang lingkup proyek : meliputi uraian masalah bisnis dan sasaran bisnis, definisi lengkap kebutuhan pemakai, profile pemakai, konsep pendekatan dan strategi mendesain solusi.
2. Struktur proyek : meliputi peran masing-masing anggota tim proyek, daftar anggota tim, struktur proyek, dan standar proses bagi anggota tim.

3. Memperkirakan resiko proyek : meliputi persiapan, mengidentifikasi resiko-resiko utama, rencana mengurangi dan mengeliminasi resiko yang telah dikenali.

2. *Planning Phase*.

Tim proyek menetapkan aplikasi apa yang akan dikembangkan dan perencanaan pembuatan solusi perangkat lunak dengan menyiapkan spesifikasi fungsional, desain perangkat lunak, rencana kerja, perkiraan biaya, penjadwalan proyek dan melakukan analisa terhadap kebutuhan bisnis, kebutuhan pemakai, kebutuhan operasi dan kebutuhan sistem. Dimana semua aktivitas tersebut akan dirangkum dalam tahapan-tahapan berikut :

a) Desain Konseptual : dimana permasalahan bisnis dilihat dari perspektif pemakai dan kebutuhan bisnis. Mendefinisikan masalah dan solusi didalam terminologi *usage scenario* (suatu sekenario pemakaian yaitu bagaimana sebuah *usecase* / fungsionalitas sistem di jalankan).
b) Desain Logikal : dimana solusi dipandang dari perspektif tim proyek dan bagaimana tim proyek mendifinisikan solusi sebagai sekumpulan layanan.
c) Desain Fisikal : dimana solusi dipandang dari perspektif *developers* (para pengembang) dan bagaimana mendefinisikan teknologi, antarmuka komponen, dan layanan dari solusi.

Milestone pada fase *planning* yaitu :

a) Validasi teknologi lengkap, meliputi produk dan teknologi yang akan digunakan untuk membuat dan menyebarkan solusi, spesifikasi *hardware* dan *software,* topologi jaringan dan konfigurasi komputer server dan komputer *client*.

b) Spesifikasi fungsional lengkap, spesifikasi ini telah di sampaikan kepada pemakai dan *stakehoder* untuk di tinjau kembali.
c) Rencana induk lengkap, dimana rencana induk merupakan kombinasi dari perencanaan dan berbagai peran-peran dalam tim.
d) Penjadwalan induk proyek lengkap, meliputi penjadwalan proyek, tanggal *release* solusi, mengkombinasikan dan mengintegrasikan informasi-informasi dari setiap peran dalam tim.
e) Aturan pengembangan dan lingkungan pengujian, meliputi lingkungan kerja untuk pengembangan dan pengujian yang tepat, menghindari hal negatif yang berdapak pada sistem.

Deliverables pada fase *planning* yaitu :

a) Spesifikasi fungsional.
b) Rencana menajemen resiko.
c) Rencana induk dan jadwal induk proyek.

3. Developing Phase

Tim proyek akan mulai membuat solusi aplikasi dengan melakukan pengkodean dan dokumentasi program, sebagai aktivitas tambahan tim juga mengembangkan infrastruktur dari solusi.

Milestone pada fase *developing* yaitu :

a) Bukti dari konsep aplikasi telah lengkap : meliputi bukti elemen kunci mengujian solusi aplikasi pada lingkungan pengujian, dan tim memimpin operasi bagi tim sendiri dan pemakai untuk memvalidasi kebutuhan mereka.
b) *Internal-build* lengkap : oleh karena solusi aplikasi di dikembangkan dalam bagian-bagian maka diperlukan sinkronisasi bagian solusi pada level produk. Frekwensi *internal-buikd* tergantung pada ukuran dan kompleksitas proyek.

Deliverables pada fase *developing* yaitu :

a) File-file source code dan file executable.
b) Skrip instalasi and seting konfigurasi untuk penyebaran.
c) Spesifikasi fungsional telah final.
d) Elemen-elemen pendukung unjuk kerja.
e) Spesifikasi pengujian dan pengujian kasus.

4. Stabilizing fase

Pada fase stabilizing tim melakukan integrasi dan pengujian beta (*test beta*) dari solusi, skenario penyebaran/distribusi solusi aplikasi, dan tim fokus bagaimana mengidentifikasi, memprioritas-kan, memecahkan persoalan sehingga solusi disiapkan untuk pelepasan (*release*).

Milestone pada fase *stabilizing* yaitu :

a) Memusatkan perhatian pada bug (*error program*).
b) Release zero-bug, dimana bug-bug yang muncul telah tertangani.
c) Kandidat *Release*, mengkomparasi sejumlah bug yang tertangani dengan zero-bug release.
d) Golden Release, ukuran kriteria keberhasilan dan zero-defect.

Deliverables pada fase *stabilizing* yaitu :

a) *Release* Final.
b) Catatan-catatan *Release*.
c) Elemen-elemen pendukung unjuk kerja.
d) Hasil test dan perangkat pengujian.
e) Kode sumber dan file *executable.*
f) Dokumen proyek.
g) Peninjauan milestone.

5. *Deployment Phase*

Pada fase ini tim proyek mulai menyebarkan/ mendistribusikan solusi teknologi, menstabilkan penyebaran, pengoperasian dan dukungan solusi, dan memperoleh persetujuan dari pemakai. Setelah penyebaran solusi, tim melakukan peninjauan proyek dan melakukan survey kepuasan pemakai.

Milestone pada fase ini yaitu :

a) Komponen-komponen utama telah tersebar.
b) Lokasi penyebaran telah lengkap.
c) Penyebaran stabil.
d) Penyebaran keseluruhan telah lengkap.

Deliverables pada fase ini yaitu :

a) Sistem informasi pengoperasian aplikasi dan dukungan yang terdiri dari prosedur dan proses, basis pengetahuan, pelaporan, *log-book*.
b) Dokumentasi tempat penyimpanan semua versi kode dan dokumen yang telah dikembangkan selama proyek.
c) Rencana training penggunaan solusi.
d) Laporan penyelesaian proyek, mencakup : versi final semua dokumen proyek, data kepuasan pelanggan, definisi tentang langkah-langkah selanjutnya.

BAGIAN 2

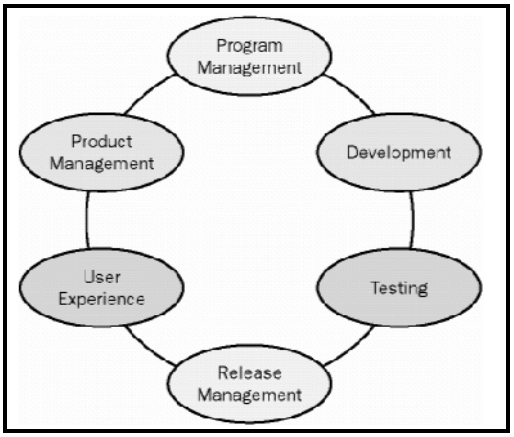
MANAJEMEN PROYEK
MODEL MICROSOFT SOLUTION FRAMEWORK

2.1. Manajemen Proyek Microsoft Solution Framework

Pengelolaan proyek perangkat lunak akan sangat tergantung dengan sumberdaya personal yang terlibat didalam tim. Dalam model MSF, peran anggota tim sangat berpengaruh bagi keberhasilan proyek perangkat lunak, karena setiap personal memiliki tanggung jawab dalam perannya masing-masing.

2.1.1. Peran Tim dalam Model MSF

Model proses MSF memiliki tahapan-tahapan yang secara garis besar telah dijelaskan pada bab sebelumnya. Dalam proyek MSF semua personal yang terlibat akan bekerja sebagai unit tunggal. Meskipun demikian masing-masing personel memiliki peran yang berbeda dan fokus pada peran masing-masing sebagai anggota tim pada setiap fase yang akan dilalui.



Gambar 7. Peran Pada Model Proses MSF.

a) *Product Management.*

Memiliki tanggung jawab yang berhubungan dengan manajemen produk perangkat lunak, termasuk juga memastikan kebutuhan dari pengguna. *product management* berkolaborasi dengan *program management* berupaya untuk menetapkan jangkauan dari proyek. Dalam menyelesaikan tujuan ini *product management* mempelajari dan menganalisa masalah-masalah bisnis yang terjadi, kebutuhan-kebutuhan bisnis, jangkauan proyek, tujuan bisnis dan profile pengguna.

b) *Program Management.*

Memiiki tanggung jawab dalam hal memastikan tujuan dari desain proyek, mendefinisikan faktor-faktor keberhasilan proyek, mendefinisikan dengan jelas konsep solusi yang akan dikembangkan, dan mengelola infrastruktur proyek.

c) *Development.*

Memberikan umpan balik kepada tim pengembang perihal implikasi teknis dari pengembangan perangkat lunak dan kemungkinan-kemungkinan pemilihan konsep solusi yang akan digunakan.

d) *User Experience.*

Menganalisa kebutuhan performansi terhadap sistem dan dukungan terhadap user dengan mempertimbangkan implikasi dari perangkat lunak untuk mempertemukan kebutuhanya.

e) *Testing.*

Memberikan umpan balik kepada tim mengenai kualitas solusi dan aktifitas yang dilakukan untuk memperoleh kualitas perangkat lunak.

f) Release Management.

Mengidentifikasi apa yang dibutuhkan dan apa yang diperlukan untuk menyebarkan solusi, bagaimana dan kapan solusi akan disebarkan, serta infrastruktur tambahan yang diperlukan.

Selain enam peran diatas, didalam tim proyek juga melingkupi stake holder proyek, akan tetapi bukan menjadi bagian dari tim inti MSF. Peran *stake holder* proyek antara lain sebagai :

a. Project Sponsor.

   Satu atau lebih individu yang menginisiasi dan menyetujui proyek dan hasilnya.

b. Customer (sponsor bisnis).

   Satu atau lebih individu yang mengharapkan untuk memperoleh nilai bisnis tertentu dari solusi aplikasi yang dihasilkan.

c. End User.

   Satu atau lebih individu yang beriteraksi langsung dengan solusi aplikasi.

d. Operations.

   Tanggung jawab organisasi yang terus-menerus mengoperasikan solusi.

2.1.2. Disiplin Model Proses MSF.

Disiplin pada model proses MSF digunakan untuk mengelola orang, proses dan teknologi. Disiplin MSF terdiri dari manajemen resiko, manajemen kesiapan, dan manajemen proyek.

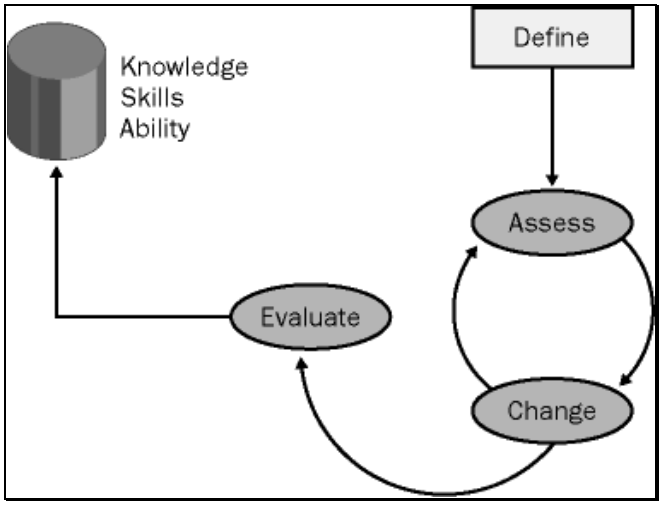
a. Manajemen Resiko.

1) Mengidentifikasi resiko : setiap individu dapat mengidentifikasi resiko yang mungkin akan menimbulkan masalah.

2) Menganalisa resiko : melakukan analisa resiko-resiko yang ada dalam format tim untuk menentukan prioritas.
3) Merencanakan resiko : menggunakan informasi yang diperoleh dari analisa resiko untuk memformulasikan strategi, rencana dan aksi.
4) Penelusuran resiko : memonitor status resiko spesifik dan mendokumentasikan kemajuan penangananya.
5) Kontrol resiko : proses menjalankan rencana penanganan resiko dan melaporkan status.
6) Pembelajaran resiko : rekaman pengetahuan untuk digunakan kembali oleh tim proyek dan perusahaan.

b. Manajemen Kesiapan.

Manajemen kesiapan membantu untuk mengembangkan pengetahuan, skill, dan kemampuan yang diperlukan untuk membuat dan mengelola proyek dan solusi.



Gambar 8. Manajemen Kesiapan Model Proses MSF.

1) *Define* : tim mengidentifikasi skenario, kemampuan, dan tingkat kecakapan yang diperlukan untuk merencanakan, membuat, dan mengelola proyek.
2) *Assess* : tim menganalisa kemampuan saat ini, yang dihubungkan dengan berbagai peran pekerjaan.
   a) Tujuan : menentukan kemampuan individu masing-masing peran.
   b) Tim membandingkan kemampuan sebelumnya dengan kemampuan saat ini untuk menjangkau tingkatan ketrampilan yang diperlukan.
3) *Change* : anggota tim mulai meningkatkan ketrampilan untuk menaikkan kecakapan yang diinginkan dengan cara :
   a) Training : pelatihan sesuai dengan rencana pembelajaran.
   b) Tracking progress : memungkinkan kesiapan individu atau secara keseluruhan dapat ditentukan setiap waktu sepanjang lifecycle.
4) *Evaluation* : tim menentukan apakah rencana pembelajaran efektif dan apakah yang telah dipelajari sukses di terapkan.

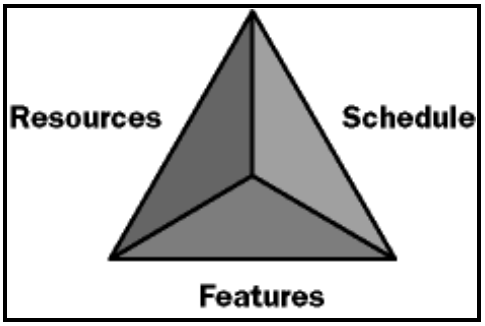
c. Manajemen Proyek.

Model MSF tidak memiliki peran manajer proyek. Fungsi-fungsi manajemen dilakukan berdasarkan enam peran dalam manajemen proyek MSF. Faktor pembeda pendekatan MSF pada manajemen proyek adalah fungsi dan aktifitas personal tidak berbentuk struktur hirarki dalam proses pengambilan keputusan. Semua anggota tim memiliki tujuan akhir yang spesifik, dimana semua tim dipertimbangkan sama pentingnya. Keputusan akhir ditetapkan berdasarkan konsensus tim inti. Namun apabila

konsensus tidak tercapai maka program manajemen yang akan membuat keputusan akhir berdasarkan atas kebutuhan pemakai dan batasan proyek.

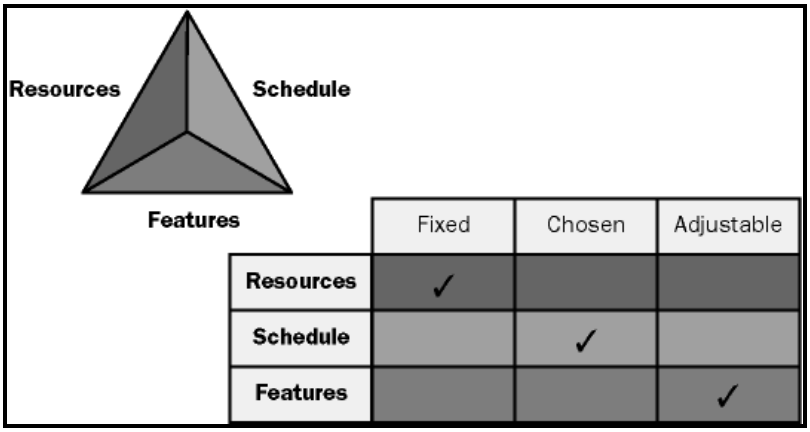
Proyek mengalami kegagalan, keterlambatan, atau melebihi anggaran yang direncanakan disebabkan karena ruang lingkup proyek rancu. Didalam ruang lingkup proyek telah ditetapkan apa yang akan dilakukan dan yang tidak akan dilakukan didalam proyek. Agar ruang lingkup proyek menjadi lebih efektif diperlukan identifikasi batasan proyek, mengatur *tradeoff*, menetapkan kontrol perubahan, memonitor kemajuan proyek.

Variabel project terdiri dari tiga komponen yaitu resource, schedule dan features yang akan ditaur dalam *tradeoff*. Perubahan yang terjadi pada suatu komponen akan berpengaruh pada komponen lainya. Agar solusi yang akan dibangun sesuai dengan kebutuhan customer diperlukan keseimbangan antara *resource* (sumber daya), schedule (penjadualan) dan features (fitur-fitur) didalam proyek. Tiga komponen tersebut dipetakan didalam apa yang disebut *tradeoff triangle*.



Gambar 10 . TradeOff Triangle.

Bagaimana menentukan pilihan prioritas tiga komponen tersebut ?. prioritas *tradeoff* ditentukan dengan memetakan tiga komponen didalam matriks *tradeoff* proyek.



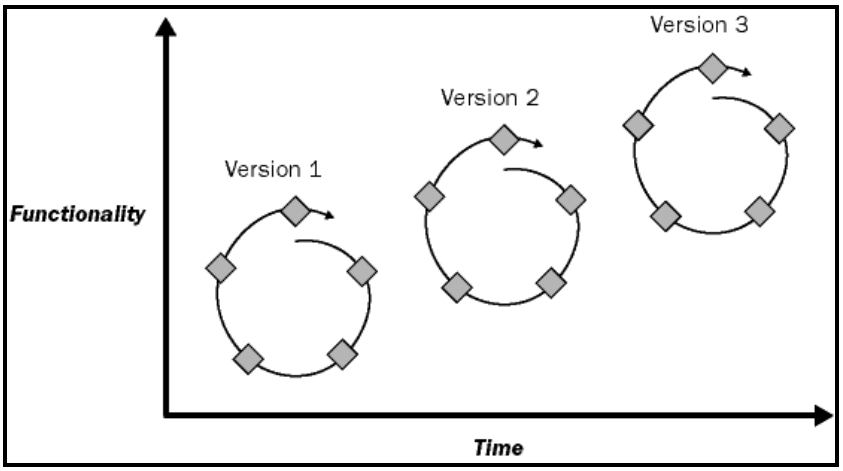
Gambar 11 . Matrik TradeOff Proyek.

Matriks *tradeoff* disusun dengan menetapkan komponen pertama, dilanjutkan dengan memilih komponen kedua dan melakukan penyesuaian terhadap komponen yang ketiga. Sehingga penyusunan matriks *tradeoff* dapat dilakukan dengan melengkapi kalimat berikut ini dengan tiga komponen *tradeoff* yang ada.

"Given fixed _____, we will choose a _____ and adjust _____ if necessary"

1) Given fixed ***resources***, we will choose a ***schedule*** and adjust the ***feature*** set if necessary."
2) Given fixed ***feature***, we will choose a ***schedule*** and adjust the set ***resources*** if necessary."
3) Given fixed ***schedule***, we will choose a ***resources*** and adjust the ***feature*** set if necessary."

Matriks *tradeoff* akan sangat bermanfaat pada proses *iterasi* (pengulangan) yang terjadi dalam perancangan solusi aplikasi, dimana setiap phase dalam model proses MSF akan terus berulang sampai tujuan pada fase tersebut telah terpenuhi dengan fitur-fitur yang terus meningkat. Iterasi mengijinkan proyek dikembangkan dalam tahapan-tahapan proyek yang lebih sederhana. Iterasi berikutnya ditentukan berdasarkan sukses atau tidaknya iterasi sebelumnya. Instan dari sebuah iterasi didalam daur hidup proyek adalah berupa release sebuah versi aplikasi, dokumentasi, build atau kompilasi secara periodik.



Gambar 10 . Fungsionalitas Release Versi.

Release versi dapat meningkatkan hubungan tim dengan customer dan memastikan ide terbaik terefleksi dalam solusi. Dalam aktivitas release versi ada beberapa petunjuk untuk memudahkan release versi :

a) Membuat rencana release versi lebih banyak.
b) Membahas iterasi dengan cepat.
c) Mengirimkan versi utama terlebih dulu.

d) Membuat fitur beresiko tinggi terlebih dulu.

Dalam sebuah proyek mungkin akan terjadi perubahan dari aspek desain dan aspek pengembangan proyek, hal ini akan muncul ketika terjadinya perubahan pada kebutuhan bisnis perusahaan. Jika terjadi hal tersebut maka diperlukan "*living document*", yaitu dokumentasi proyek oleh karena proyek telah mengalami perubahan.

Model MSF merekomendasikan untuk melakukan persiapan dalam melakukan *build* (kompilasi) secara teratur untuk dapat memastikan solusi aplikasi telah stabil dan telah teruji secara akumulatif dengan sekumpulan data test sebelum *release* (aplikasi siap di implementasikan). Kompilasi secara teratur juga untuk memastikan semua kode program *compatible* dan mengijinkan subtim untuk melanjutkan pengembangan dan pengujian secara berulang-ulang.

BAGIAN 3

MENGUMPULKAN DAN MENGANALISA INFORMASI

2.1. Mengumpulkan Informasi.

Informasi menjadi bagian yang sangat penting dalam perancangan solusi. Ketika mengumpulkan informasi perlu diperhatikan berbagai tipe dan karakteristik informasi untuk memastikan informasi yang diperoleh sesuai dengan kebutuhan dalam pengembagan perangkat lunak. Arsitektur perusahaan adalah representasi suatu bisnis tertentu dan suatu sistem yang dinamis. Untuk memperoleh informasi perihal arsitektur perusahaan pertimbangkan empat kategori informasi berikut ini untuk dianalisa lebih lanjut.

1. Informasi Bisnis.

   Informasi bisnis menjelaskan bagaimana sebuah proses bisnis berjalan yang akan menguraikan aktifitas fungsional di dalam organisasi. Selain itu menguraikan sasaran bisnis, tujuan bisnis, produk dan jasa serta fungsi keuangan perusahaan, integrasi fungsi dan proses bisnis, struktur organisasi dan interaksi antar elemen organisasi. Yang tidak kalah penting adalah strategi dan rencana bisnis agar organisasi terus berkembang.

2. Informasi Aplikasi.

   Informasi aplikasi meliputi layanan dan fungsi organisasi yang berhubungan dengan kemampuan dan fungsi personal dalam organisasi untuk mencapai suatu tujuan bisnis. Informasi aplikasi

menguraikan layanan terotomatisasi atau yang non otomatisasi untuk mendukung proses bisnis dari sisi interaksi dan ketergantungan internal pada sistem aplikasi.

3. Informasi Operasi.

   Informasi operasi menguraikan informasi apa yang harus diketahui organisasi untuk menjalankan proses bisnis dan operasinya meliputi : model data standar, kebijakan manajemen data, diskripsi pola konsumsi informasi dan produksi dalam bisnis. Selain itu hubungan antar proses-proses bisnis dan informasi yang dibutuhkan untuk menjalankan proses bisnis dan menetapkan standar dalam membuat, memperoleh kembali, memperbaharui, dan penghapusan data atau informasi.

4. Informasi Teknologi.

   Informasi teknologi merupakan definisi layanan teknis yang dibutuhkan untuk pelaksanaan dan dukungan terhadap misi bisnis, meliputi : topologi jaringan, lingkungan pengembangan, antarmuka pemrograman aplikasi, keamanan, layanan jaringan, layanan database manajemen sistem (DBMS), spesifikasi teknis, lapisan hardware, sistem operasi. Informasi teknologi ini dapat digunakan untuk menetapkan standar antarmuka, layanan dan model apliasi yang digunakan dalam pengembangan solusi.

2.2. Teknik Mengumpulkan Informasi.

Agar informasi yang dikumpulkan menjadi efektif maka diperlukan teknik-teknik pengumpulan informasi. Teknik mengumpulkan informasi dapat dilakukan dengan :

1. Shadowing, aktivitas-aktivitas dalam kegiatan ini meliputi :

a) Mengamati user dalam melakukan tugas dilingkungan pekerjaan.
b) Bertanya kepada user tentang tugas yang dikerjakanya.
c) Memahami tujuan user melakukan tugasnya.
d) Shadowing dapat dikategorikan pada aktif dan pasif.
   - Pasif shadowing : pengamatan terhadap karyawan dan mendengarkan semua penjelasanya.
   - Aktif shadowing : bertanya kepada karyawan untuk menjelaskan tentang kegiatan dan aktifitasnya.
e) Shadowing sesuai untuk tugas-tugas kayawan yang dilakukan secara teratur dalam melakukan pekerjaanya.
f) Shadowing tidak cocok diterapkan untuk karyawan bagian akunting, pengembang, dan manajemen.
g) Pertanyaan-pertanyaan dalam shadowing misalnya :
   - Bagaimana karyawan melakukan struktur pekerjaanya ?
   - Apa yang dilakukan ketika memulai atau menyelesaikan tugasnya ?
   - Bagaimana "sistem" bertentangan dengan aktivitas pekerjaan karyawan ?
   - Bagaimana gangguan berpengaruh pada karyawan, dan apakah dapat memulai pekerjaanya lagi setelah disela dengan aktivitas lain?
   - Dengan berapa orang karyawan berinteraksi selama melakukan aktifitas pekerjaanya ?
   - Bagaimana karyawan menggunakan cara untuk memudahkan dalam menyelesaikan tugasnya ?

- Berapa macam keragaman karyawan dalam menyelesaikan tugasnya ?

h) Informasi-informasi yang dibutuhkan dalam shadowing diantaranya :
   - Bagaimana karyawan saat ini melaksanakan tugasnya ?
   - Bagaimana proses pekerjaan dibuat lebih efisien ?
   - Terkait dengan tugas yang mana yang mungkin mempengaruhi perancangan solusi ?
   - Apa saja fitur sistem yang diperlukan untuk mendukung tugas karyawan.
   - Apa saja yang merupakan kriteria unjuk kerja ?
   - Bagaimana fitur solusi tersusun ?
   - Bagaimana sistem yang sekarang dapat ditingkatkan ?

2. Interview.
   a) Interview sesuai untuk aktifitas pekerjaan yang dilakukan dalam jangka waktu tertentu (mingguan, bulanan atau tahunan) atau proses yang memerlukan sedikit sekali atau tidak samasakali adanya intervensi karyawan, sebagai contohnya layanan sistem pembayaran otomatis.
   b) Interfiew adalah aktifitas bertemu satu-satu antara anggota tim dengan user atau stakeholder.
   c) Kualitas informasi tergantung pada interviewer dan interviewee.
   d) Point-point penting dalam melakukan interfiew adalah :
      - Dimulai dengan daftar pertanyaan yang tidak spesifik dan mendorong *interviewee* memikirkan tentang tugas yang dilakukannya.

- Tanyakan tentang urutan tugas “besar” yang dilakukanya dan bagaimana karyawan memecahnya menjadi tugas-tugas yang lebih “kecil”.
- Tanyakan kepada interviewee untuk mengidentifikasi informasi yang pada umumnya hilang dan alur alternatif memperoleh informasi.
- Ulangi pertanyaan-pertanyaan terdahulu beberapa kali.
- Tanyakan tentang gagasan karyawan mengenai solusi, tetapi hindari anggapan bahwa gagasan itu adalah solusi yang benar.

e) Contoh pertanyaan-pertanyaan untuk interfiew :
- Masalah apa yang anda hadapi selama mengerjakan tugas ?
- Bantuan apa yang anda butuhkan dari orang lain ?
- Apakah anda memiliki kebutuhan yang tidak didokumentasikan ?
- Kebijakan apa yang dapat membantu atau menghalangi anda melaksanakan tugas ?
- Individu dan dokumen apa yang diperlukan untuk mengerjakan tugas ?
- Pemakai lain atau sistem apa yang mempengaruhi pekerjaan anda ?

3. Focus Group.
   a) Seperti teknik interview akan tetapi fokus pada kelompok bukan individu.
   b) Teknik ini digunakan jika dalam proses pengumpulan informasi memerlukan banyak karyawan.
   c) Memastikan siapa yang berpartisipasi didalam proses bisnis.

d) Mendifinisikan topik dan mengajak kelompok fokus pada topik tertentu.
e) Individu dalam fokus group dapat saling mengisi gap pengetahuan satu dengan lainya dan menyediakan diskripsi lengkap proses bisnis.

4. Survey
   a) Survey dilakukan dengan membuat daftar pertanyaan untuk mengumpulkan informasi dari responden.
   b) Dibutuhkan pelatih untuk membuat pertanyaan survei yang relefan dan melakukan analisis terhadap hasil survei.
   c) Keuntunganya adalah user dapat memberikan jawaban tanpa membubuhkan identitas, yang mustahil dapat dilakukan dengan teknik lain.
   d) Perlu melakukan penghalusan bahasa agar lebih netral sehingga identitas responden terjaga.
   e) Informasi hasil survay meliputi :
      - Struktur organisasi, kebijakan organisasi, dan aktivitas yang memberikan kemudahan atau justru bertentangan dengan tugas-tugas karyawan.
      - Frustasi dengan struktur dukungan teknis atau kebijakan yang ada.
      - Kebutuhan khusus yang berhubungan dengan hardware dan software.
      - Isu pelatihan, efektifitas program training saat ini, program pelatihan yang disukai user, program pelatihan yang berjalan dengan baik di lingkungan kerja.

5. Instruksi User.
    f) Mengijinkan tim pengembang terlibat dengan aktifitas user dan melihat proses dari perspektif pemakai.
    g) Tim mungkin akan memperoleh pengetahuan baru dari karyawan yang tidak tersedia pada sistem.
    h) Metode ini mungkin akan membuat user frustasi jika user tidak terbiasa mengajar orang lain.
    i) Adanya perbedaan personal dalam melaksanakan tugas yang sama dengan cara yang berbeda.
    j) Melalui help-desk tim akan memperoleh informasi tentang pengalaman karyawan. Help-desk merupakan sumberdaya yang sangat relefan karena dapat memberikan perspektif langsung pada pengalaman karyawan.
    k) Dengan mempersilahkan karyawan untuk mengajarkan tugasnya, maka tim dapat mengumpulkan informasi untuk menentukan :
        - Desain antarmuka user
        - Kebutuhan training untuk aktifitas saat ini dan yang akan datang.
        - Kriteria unjuk kerja sistem.
        - Pengaruh lingkungan fisik pada suatu tugas.

2.3. Menganalisa Informasi

Aktifitas memperoleh dan menganalisa informasi merupakan aktivitas yang terus berulang. Informasi-informasi dalam jumlah besar yang diperoleh perlu di tinjau ulang apakah informasi tersebut

relefan dengan kebutuhan bisnis. Tim akan mensitesis (memadukan) informasi untuk mendiskripsikan kondisi bisnis saat ini.

Setelah aktivitas sintesis informasi selanjutnya tim membuat *usecase* dan *usage scenario* untuk mendokumentasikan proses bisnis dan kebutuhan pemakai. Usecase dan usage scenario menyediakan struktur bagi tim pengembang dalam mendesain perangkat lunak (solusi).

Usecase menggambarkan fungsionalitas dari sistem dan bagaimana personal (aktor) berinteraksi dengan sistem untuk mendapatkan sebuah nilai tertentu. Kegunaan dari usecase adalah :

- Mengidentifikasi proses bisnis dan semua aktivitas dari awal sampai selesai.
- Mendokumentasi konteks dan isu lingkungan.
- Memastikan hubungan antara kebutuhan bisnis dan kebutuhan pemakai.
- Menjelaskan kebutuhan-kebutuhan didalam konteks penggunaanya.
- Fokus kepada user dan tim pengembang.

Usage scenario menggambarkan interaksi antara pemakai dengan sistem dan menyediakan informasi tambahan tentang aktivitas dan urutan tugas dari sebuah proses. *Usage Scenario* dan *use case* bersama-sama mendeskripikan aliran proses. Langkah-langkah membuat *usecase* dan *requirement* (kebutuhan) dapat diuraikan sebagai berikut :

1. Memperoleh dan menganalisa informasi melalui interview.
2. Menentukan rancangan awal usecase dari hasil interview.

3. Membuat draft requirement yang diperoleh dari hasil interview dan usecase.
4. Membuat dokumentasi internal yang berisi katalog aktor, aturan bisnis dan daftar istilah teknis.

Sebagai contoh, studi kasus pada perusahaan sepeda ”Adventure Works Cycle” akan mengembangkan sistem otomasi penjualan melalui website.

1) Memperoleh dan menganalisa informasi melalui interview.

Contoh ringkasan hasil interview tim dengan *Manajer Penjualan Wilayah* :

Pada tahun ini penjualan kami sangat bagus. Pekerjaan kami menjadi semakin banyak kemudahan setelah perusahaan melengkapi kami dengan laptop dan PDA. Departemen penjualan menetapkan tiga tujuan pada tahun fiskal mendatang, yaitu fokus pada pelanggan, menggunakan waktu staff penjualan lebih efektif, dan mengelola peluang penjualan lebih baik.

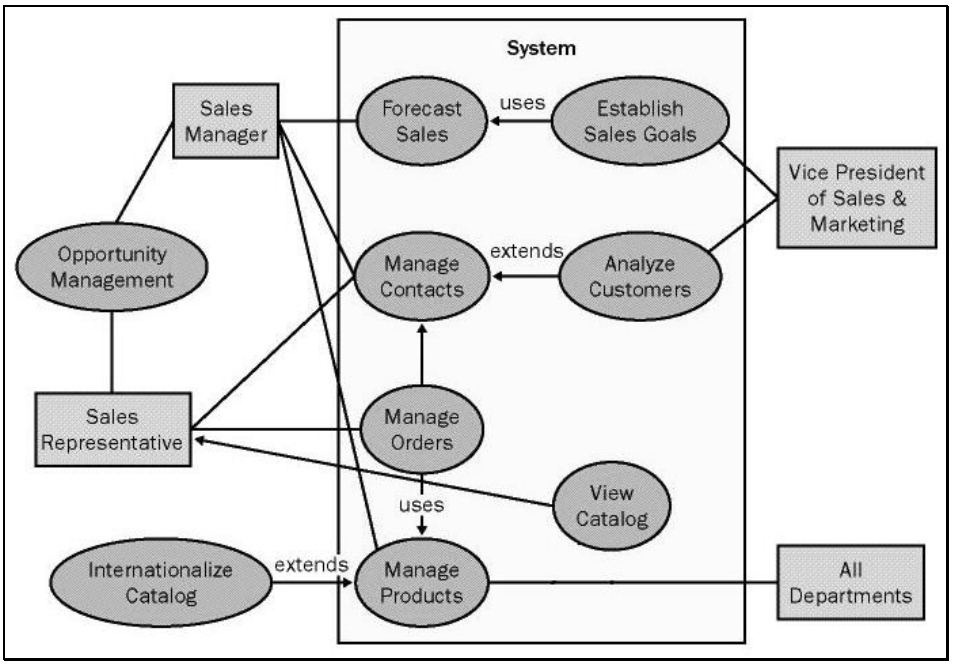
Kami tidak mudah menganalisa basis pelanggan kami, sekarang ini data pelanggan tersimpan di sistem kami, tetapi tidak mudah untuk mengakses kembali data dan melihat data dengan cara yang berbeda. Untuk dapat mengakses data pelanggan terbaik kami dan bagaimana mereka menjadi pelanggan terbaik, kami memerlukan cara untuk mengakses data dan dapat menganalisisnya dengan sangat berarti.

Masalah lainya adalah kami memiliki kemampuan dalam mepresentasikan produk kami dan mencapai sukses penjualan di pasaran luar negeri. Sekarang ini bahasa semua informasi pada komputer kami ditulis hanya dalam bahasa inggris. Kami

membutuhkan informasi dalam multibahasa dan multidaerah dibandingkan harus menterjemahkan informasi.

Tim kami memerlukan untuk memperoleh penentuan harga terakhir, saat ini sales representative harus melakukan koneksi ke jaringan korporat dan mendownload daftar harga terbaru setiap hari, proses download tidak dapat mengidentifikasi dimana harga telah disesuaikan untuk sales secara individu. Sehingga sales harus mendownload keseluruhan daftar harga setiap pagi hari.

2) Menentukan rancangan awal usecase dari hasil interview.



Gambar 11. Rancangan Awal Diagran *Usecase*.

Berikut ini adalah contoh *Usage Scenario* dari sebuah *Usecase* dengan nama "*Memesan Spesifikasi Produk*". *Usecase* ini menggambarkan bagaimana pelanggan (*customer*) memesan spesifikasi produk melalui internet.

Use case title	: Memesan Spesifikasi Produk
Abbreviated title	: Memesan Spesifikasi Produk
Use case ID	: UC 05.1
Requirements ID	: 15.1
Intent	: Untuk menyediakan spesifikasi produk lengkap kepada pelanggan.
Scenario narrative	: Pada saat pelanggan melihat item dalam katalog, pelanggan meminta spesifikasi-spesifikasi produk.
Actors	: Pelanggan
Preconditions	: Pelanggan membuka katalog produk. Pelanggan melihat item didalam katalog .
Basic course	: 1. *Usecase* dimulai ketika pelanggan meng-klik Pemesanan Spesifikasi Produk untuk item yang dipilih. 2. Pelanggan memilih format dari spesifikasi produk. 3. Pelanggan memilih cara pengiriman produk. 4. Alamat pelanggan telah dikonfirmasi. 5. *Usecase* berakhir ketika pemesanan telah selesai, terkirim dan siap untuk menunggu kiriman katalog.
Alternative course	: -
Postconditions	: Pemesanan telah selesai, terkirim dan siap untuk menunggu kiriman katalog. Pelanggan kembali pada katalog yang dibuka terakhir kali.
Uses/extends	: Pelanggan membaharui profile, pelanggan membuat profile baru, dan mengirimkan pemesanan.
User implementation requests : Tidak ada	
Frequency	: Terjadi antara 17-100 sesi pelanggan di website.
Unresolved issues	: Tidak ada
Authority	: Mike Danseglio
Modification history **Author** **Description**	: 6 November 2006 : Heidi Steen : Versi Inisial

3) Membuat draft requirement yang diperoleh dari hasil interview dan usecase.

Req ID	Req Description	Priority	Source	Questions
1	Dapat melakukan sinkronisasi dengan aplikasi online kita untuk memperoleh seluruh informasi sepanjang hari	1	Manager penjualan wilayah	- Informasi spesifik apa yang dihimpun ?, - Apa yang dimaksud sinkronisasi ?, dengan apa ?
2	Memperoleh kembali data pelanggan dan mengijinkan kami melihat data dengan cara yang berbeda	1	Manager penjualan wilayah	- Dengan jalan bagaimana anda melihat data ? - Apa yang dimaksud data pelanggan ?

4) Membuat dokumentasi internal :

Katalog aktor.

Actor	Description	Source	Business Title
Consumer	Melihat-lihat dan memesan produk online, mencetak spesifikasi produk	Manager penjualan wilayah	Web customer, sales representative, sales manager, reseller, customer, consumer.
Customer	Mengacu kepada Consumer, Client, atau Reseller.	Manager penjualan wilayah	Web customer, reseller, consumer, customer, contract client.

Aturan Bisnis.

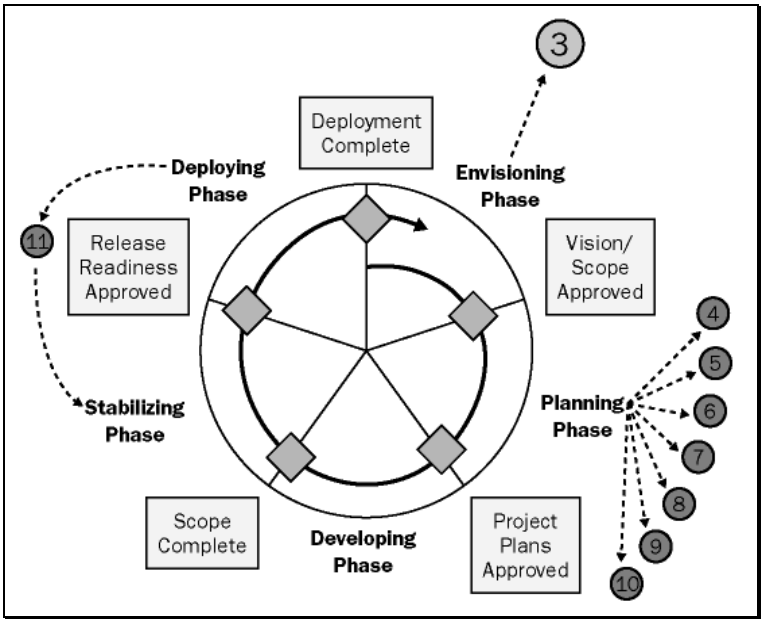
BR ID	Business Rule Title	Description	Authority	UC/BR	Current Process
1	Diskon	Sales representatif dapat memberikan diskon 15% - 20% dengan otorisasi manager sales	Territory Sales Manager	**UC 05.5.1** **UC 05.5.1.1**	1. Sales representative melakukan negosiasi diskon dengan pelanggan. 2. Jika diskon <= 15% sales representatif dapat langsung menyetujui. 3. Jika diskon 15%-20%, sales representative mengirim e-mail sales manager untuk meminta diskon.

BAGIAN 4

TAHAP *ENVISIONING*

4.1. Membuat Dokumen Visi/Scope

Pada fase ini tim meninjau permasalahan bisnis yang akan di pecahkan dan bagaimana permasalahan tersebut berhubungan dengan bisnis perusahaan, pemakai, dan lingkungan yang dapat membantu tim memperoleh pandangan yang jelas bagaimana tim akan menyelesaikan permasalahan bisnis yang ada.



Gambar 12. Tahap Envisioning.

Dokumen visi/scope merepresentasikan kesepakatan awal diantara semua orang yang terlibat didalam proyek. Dokumen ini akan membantu tim untuk mencapai tujuan bisnis yang utama.

Seteleh proyek disetujui tim menggunakan dokumen ini untuk membuat perencanaan proyek. Untuk membuat dokumen visi/scope tim melakukan interview dengan pemakai dan *stakeholder*, melakukan analisa terhadap fungsionalitas sistem (*usecase*), dan mengidentifikasi asumsi-asumsi serta batasan didalam organisasi. Dokumen visi/scope haruslah fokus dalam memahami dan bagaimana menifinisikan permasalahan, isi dari dokumen ini meliputi :

1) Statemen Permasalahan.

Statemen permasalahan biasanya berisi deskripsi harapan-harapan yang ingin dicapai dalam aktivitas bisnis. Karena tujuan dari setiap proyek adalah memecahkan masalah, maka memahami permasalahan akan menentukan desain solusi yang akan dibuat. Sehingga statemen permasalahan haruslah memiliki informasi yang cukup perihal permasalahan bisnis. Berikut ini contoh-contoh statemen permasalahan bisnis :

- Operator telepon tidak dapat menangani panggilan dalam jumlah banyak karena waktu yang mereka gunakan selain untuk menjawab telepon juga untuk beriteraksi dengan aplikasi secara bergantian.
- Pemakai memerlukan arahan yang jelas dari sistem untuk memecahkan masalah ketika terjadi.
- Kami ingin meningkatkan registrasi *user* secara online dengan merancang website kami sehingga mudah untuk di eksplorasi.

2) Statemen Visi

Statemen visi berisi kalimat singkat yang cukup mudah diingat, dipahami, dan cukup kuat memotivasi. Statemen visi memiliki

karakteristik yang pesifik, terukur, dapat dicapai, relefan dan berbasiskan waktu. Berikut ini contoh-contoh statemen visi :

- Sebelum akhir tahun ini, kami akan menjadi perusahaan dengan pendapatan tertinggi dengan meningkatkan penjual online kami.
- Selama tahun fiskal ini, kami akan membuat semua pelanggan dapat memberikan *bookmark* untuk memilih halaman dari website kami untuk diakses dari komputer atau perangkat manapun yang digunakanya.

3) Profile Pemakai

Profile pemakai berguna untuk mengidentifikasi pemakai dimana tim proyek dapat dapat menilai perkiraan, resiko, tujuan dan batasan proyek. Dalam membuat profile pemakai perlu mempertimbangkan : bagaimana pemakai berinteraksi dengan solusi, memahami faktor-faktor yang berhubungan dengan kemampuan menggunakan solusi termasuk *hardware* dan *software-nya*, informasi tentang kesulitan user terhadap solusi yang sama, kultur dan kebutuhan lokal pemakai, batasan secara geografis seperti lokasi dan jumlah pemakai, aliran informasi diantara pemakai, fungsi-fungsi user, komunikasi organisasi, dan kebijakan pengambilan keputusan.

4) Jangkauan Proyek

Salah satu faktor kritis keberhasilan proyek berhubungan dengan definisi jangkauan proyek yang jelas sehingga apasaja yang akan dikerjakan dan apasaja yang tidak dikerjakan didalam proyek. Jangkauan proyek berhubungan dengan fitur solusi yang akan di kembangkan. Untuk memulai mendefinisikan jangkauan proyek yaitu dengan menentukan fungsionalitas sistem (*usecase*) yang memiliki prioritas tinggi. Seperti pada gambar 11, *usecase* yang berada

didalam kotak adalah *usecase* yang akan direalisasikan, sementara usecase yang berada diluar kotak tidak termasuk dalam lingkup proyek.

Aktivitas lain yang cukup penting untuk mendefinisikan jangkauan proyek adalah menyaring kebutuhan. Dengan mempertimbangkan kebutuhan level tinggi berikut ini :

"Sistem harus mendukung kebutuhan lokal untuk mempertemukan kultur bagi pemakai akhir"

Setelah menyaring kebutuhan level tinggi diatas selama fase envisioning, maka kebutuhan akan menjadi :

"Sistem harus memiliki prosedur, untuk globalisasi, lokalisasi dan kemampuan akses"

Selanjutnya tim proyek menetapkan *tradeoff triangle* dimana tim akan menyeimbangkan variabel proyek seperti sumberdaya (*resource*), fitur-fitur (*features*), dan penjadualan (*schedule*). Tim juga harus mendefinisikan versi solusi yang akan di kembangkan, asumsi-asumsi seperti misalnya :

- Kami akan menggunakan Microsoft.Net Framework.
- Kami akan bekerja dengan dua tim pengembang.
- Kami akan menerapkan sistem OLAP (*on line analitycal processing*) dan OLTP (on line transactional processing). OLAP dan OLTP digunakan untuk aktivitas *datawarehousing* dan *datamining*.

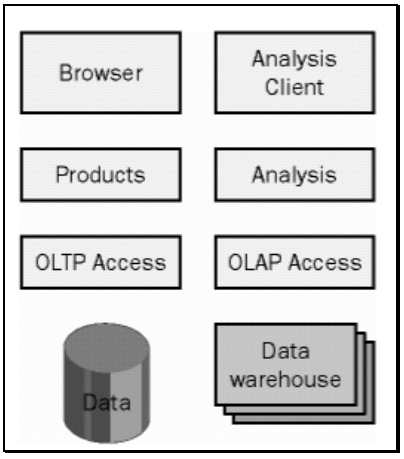
Tim juga mendefinisikan batasan-batasan proyek yang meliputi :

- Penganggaran
- Karakteristik sistem dukungan
- Arsitektur sistem jaringan

- Kebutuhan keamanan
- Sistem operasi
- Peningkatan teknologi secara terencana
- Lebar data (*bandwith*) jaringan
- Kesepakatan dukungan dan perawatan
- Tingkatan pengetahuan staf pengembang dan staf pendukung
- Pelatihan bagi pemakai
- Estimasi-estimasi yang meliputi waktu, usaha dan biaya.

5) Konsep Solusi

Konsep solusi adalah pendekatan yang digunakan tim untuk mempertemukan tujuan dari proyek. Konsep solusi menjelaskan terminologi-terminologi umum bagaimana tim mempertemukan kebutuhan proyek. Contoh konsep solusi sebagai berikut :



Gambar 13. Konsep Solusi.

6) Sasaran Proyek

Sasaran proyek terdiri dari dua macam yaitu sasaran bisnis dan sasaran desain. Sasaran bisnis merepresentasikan apa yang diharapkan oleh pemakai dimana solusi yang dihasilkan mendukung

kebutuhan bisnis organisasi. Sasaran bisnis untuk proyek pengembangan e-commerce, misalnya :

- Memperluas pasar perusahaan melebihi jangkauan pasar saat ini.
- Mempercepat waktu penjualan produk melalui website.

Sedangkan sasaran desain fokus pada atribut-atribut solusi (menyangkut periha yang bersifat teknis). Sasaran desain untuk proyek pengembangan e-commerce, misalnya :

- Meningkatkan layanan kepada pelanggan dengan menurunkan waktu tunggu download dibawah lima detik.
- Membatasi ketergantungan konektivitas pada server.

4.2. Dokumen Struktur Proyek.

Definisi dari pendekatan tim untuk mengorganisasikan dan mengatur proyek mendeskripsikan

- Struktur administratif
- Standar dan proses
- Sumberdaya proyek dan batasan proyek

Jika diharapkan adanya perubahan dalam langkah-langkah proyek kedepan, Maka tim perlu menguraikan secara detail bagaimana tim menangani perubahan tersebut. Dibawah ini adalah contoh dokumen change manajemen :

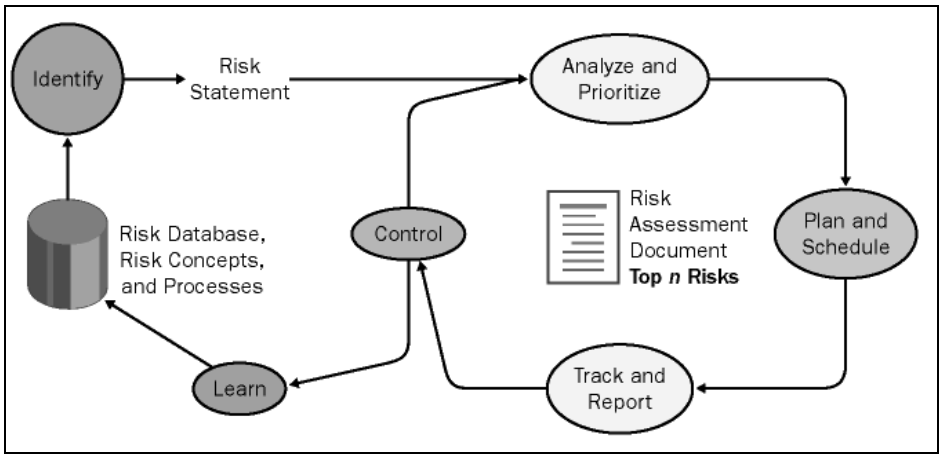
Change Managemen
Prioritas tertinggi proyek bernama scout adalah mengirimkan fitur awal yang ditetapkan berdasarkan tanggal penyelesaian proyek. Perkiraan penyelesaian pembuatan website tanggal 1 september 2007, dan perkiraan penyelesaian untuk otomatisasi sales tanggal 15 Nov 2007.

Proses kontrol perubahan dan kepemilikan dokumentasi Program Manager bertanggungjawab mengenai perubahan kontrol proses dan dokumen. Manager proyek aplikasi *Adventure Work Cycle* akan menjadi pembuat keputusan utama didalam proses kontrol perubahan untuk mengelola perubahan yang diminta customer. ***Fitur berdasarkan versi*** Setiap fitur didentifikasi dengan Critical v 1.0 want v 1.0, critical v 2.0 want v 2.0
Perubahan badan penasehat Perubahan badan penasehat dibuat oleh Contoso Ltd. (sebagai development team dan program manager), Adv. Work Cycle (sebagai manager proyek dan pengembang), keduanya dapat membuat permintaan perubahan fitur. Anggota tim yang lain dapat mengusulkan fitur secara individu, siapa yang kemudian dapat menambahkan usulan dalam daftar diskusi pada pertemuan berikutnya. Jika secara individu menganggap suatu fitur *critical*, individu dapat meminta tim melakukan evaluasi pada diskusi berikutnya.
Evaluasi fitur Definisi fitur dan pengaruhnya pada desain dan solusi, versi 1.0 atau versi lainya serta resiko akan dievaluasi oleh tim.
Tradeoff Fitur Fitur-fitur di komparasi dengan fitur-fitur lainya dan tradeoff yang mungkin akan ditetapkan.

Tradeoff resource Tradeoff triangle akan direview dan elemen yang sesuai akan ditambahkan atau dihilangkan
Fitur terpecahkan untuk desain Setelah fitur critical terpecahkan (v.1.0, 2.0, 3.0) atau want (v.1.0, 2.0, 3.0) atau sumber daya ditambahkan atau fitur lain dihilangkan atau dikurangi, sehingga fitur akan memasuki point proses desain. Penganggaran dan kontrak akan disesuaikan. (jika Adventure Works Cycle memutuskan bahwa penjadwalan di sesuaikan dengan fitur, mereka dapat meminta perubahan jadual, dan perubahan lain seperti kontrak, desain, dan anggaran)
Perubahan Cuttoff Kedua perusahaan setuju ketika scope mencapai milestone 70%, fitur-fitur baru diperkenalkan (v2,v3)

4.3. Manajemen Resiko.

Sebuah risiko timbul akibat ketidaktentuan keputusan dan hasil. Aspek yang penting dalam pengembangan solusi yang berhasil adalah pengontrolan dan peringanan resiko yang telah melekat/menjadi sifat suatu proyek. Manajemen resiko MSF menerapkan pendekatan proaktif didalam aktivitas proyek. Tim akan terus melakukan *assesment* apa saja yang dapat menyebabkan kesalahan dan bagaimana tindakan secara preventif atau meminimalkan kesalahan yang terjadi.



Gambar 14. Konsep Solusi.

Manajemen resiko MSF mendifinisikan enam langkah dalam menajemen resiko, dimana langkah-langkah manajemen resiko adalah :

1) Identifikasi resiko, mengidentifikasi resiko dan membuat tim sadar akan masalah-masalah yang potensial terjadi.
2) Analisa resiko dan menentukan prioritas, menterjemahkan informasi dan data resiko untuk memberi informasi kepada tim untuk membuat keputusan dan memprioritaskan sumberdaya untuk meringankan resiko.
3) Merencanankan dan menjadualkan resiko, menggunakan hasil analisa resiko untuk menentukan strategi peringanan resiko, rencana dan aksi.
4) Penelusuran dan pelaporan resiko, memonitor progres rencana peringanan resiko dan memberi hasil ke tim proyek, pemakai, dan stakeholder.
5) Kontrol resiko, menjalankan rencana peringanan dan melaporkan status resiko kepada tim dan pemakai.

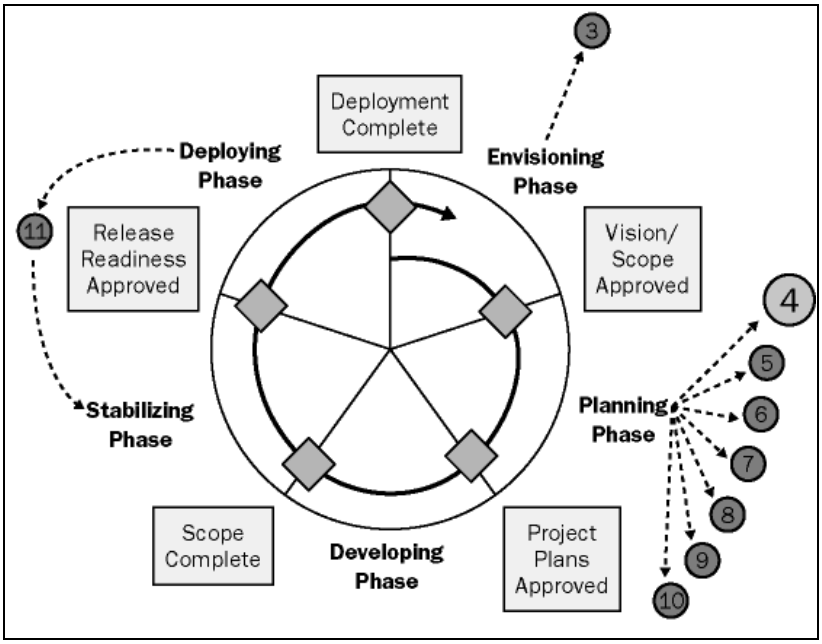
6) Pembelajaran resiko, dokumen pembelajaran hasil proyek sehingga tim dan organisasi dapat menggunakan informasi ini.

BAGIAN 5

TAHAP PLANNING

5.1. Proses Tahap *Planning.*

Pada fase *planning* tim akan mengerjakan tiga macam desain yaitu desain konsep, desain logik, dan desain fisik. Desain konsep dimulai sejak fase *envisioning* dan dilanjutkan pada fase *planning*. Karena MSF menerapkan model *iterative* (proses berulang) maka desain konsep akan menjadi pondasi bagi proses desain selanjutnya yaitu desain logik dan desain fisik.



Gambar 15. Fase *Planning*.

Tujuan dari desain konsep adalah memahami permasalahan bisnis yang akan dipecahkan, memahami kebutuhan-kebutuhan bisnis, kebutuhan pemakai, dan pengguna akhir, dan tujuan yang

terakhir adalah menggambarkan target kondisi bisnis yang akan datang bagi perusahaan.

5.1.1. Desain Konseptual.

Melakukan sintesis terhadap informasi yang diperoleh yang menghasilkan model informasi seperti hubungan antara proses bisnis, sistem bisnis, para pemakai, aliran proses dan urutan aktivitas. Selain itu profile pemakai yang telah terbaharukan, kandidat kebutuhan-kebutuhan (*requirements*) dan *usecase* yang lebih detail. Setelah melakukan sintesis terhadap informasi, aktivitas selanjutnya adalah membuat *usage scenario*. Kebutuhan-kebutuhan disini akan dibagi menjadi empat macam diantaranya adalah kebutuhan pemakai, kebutuhan sistem, kebutuhan operasi dan kebutuhan bisnis.

Scope pada desain konseptual meliputi memahami masalah bisnis yang akan dipecahkan, memahai kebutuhan bisnis, kebutuhan pelanggan dan kebutuhan pemakai, dan menggambarkan target status bisnis dimasa yang akan datang. Pada tahap ini tim tidak menyelesaiakan spesifikasi fungsional akan tetapi baru memulai membuat spesifikasi fungsional, tidak mendifinisikan komponen sistem tetapi mengidentifikasi masalah bisnis yang akan dibuat komponenya, pada tahap ini juga tidak memilih solusi teknologi tetapi mencatat aktivitas bisnis dan menggambarkan batasan dan relasi diantara aktivitas-aktivitas bisnis.

Spesifikasi fungsional merupakan gudang virtual dari sebuah proyek dan desain artifak-artifak selama fase *planning*. Artifak adalah hasil dari aktivitas desain selama melakukan desain konseptual, logikal dan fisikal yang berupa model UML seperti *usecase* diagram,

usage scenario, kandidat *requirement*, kandidat fitur don model informasi lainya. Elemen-elemen dari spesifikasi fungsi adalah :

1. Ringkasan desain konseptual, dengan artifak berupa :
   - Usecase, usage scenario
   - Konteks model sebagai gambaran sistem dan user manual yang ada saat ini.
2. Ringkasan desain logikal, dengan artifak berupa :
   - Model tugas dan task-sequence.
   - Objek logikal dan model layanan.
   - Model konseptual solusi
   - Aliran *screen user interface*
   - Model database logikal
   - Arsitektur sistem
3. Ringkasan desain fisikal
   - Pemaketan komponen
   - Topologi distribusi komponen
   - Panduan penggunaan teknologi
   - Arsitektur infrastruktur dan desain
   - Deskripsi dari tampilan user interface
   - Model database fisikal
4. Standarisasi dan proses, yang menjadi panduan bagi tim proyek untuk melaksanakan tugas-tugas selama proyek.

Tahapan-tahapan didalam desain konsep adalah *research, analysis* dan *optimation.*

1. Pada tahap *research* tugas-tugas yang dilakukan adalah :
   - Memperoleh jawaban atas pertanyaan inti.
   - Mengidentifikasi inti proses bisnis dan aktivitasnya.

- Memprioritaskan aktivitas dan proses.
- Memvalidasi, menyaring, dan memperluas draft kebutuhan, use case, dan *usage scenario*.

2. Pada tahap *analysis* tugas-tugas yang dilakukan adalah :
   - Meninjau hasil riset bisnis dan riset pemakai.
   - Menyaring kandidat *requirement*.
   - Mendokumentasikan dan memodelkan konteks, *workflow, task sequence* dan relasi yang berhubungan dengan lingkungan.
3. Pada tahap *optimation* tugas-tugas yang dilakukan adalah :
   - Mengoptimalkan konsep solusi.
   - Memvalidasi dan mengetes proses bisnis yang telah dikembangkan.

Contoh Draft Statemen *Requirement* pada fase *Envisioning*.	
Req ID	**Requirement**
1	Identifikasi pelanggan berdasarkan produk dan lokasi.
2	Identifikasi penurunan penjualan pada pelanggan.
3	Identifikasi pelanggan potensial.
4	Identifikasi pembeli terbanyak.

Tabel 1. Draf *Requirement* pada Fase *Envisioning.*

Contoh *Restate Requirement* pada fase *Planning*.	
Req ID	**Requirement**
1.1	Haruslah dapat menganalisa data pelanggan.
1.1.1	Haruslah dapat menganalisa keuntungan berdasarkan produk.
1.1.2	Haruslah dapat menganalisa keuntungan berdasarkan pelanggan.
1.1.3	Haruslah dapat menganalisa keuntungan berdasarkan daerah.

Contoh *Restate Requirement* pada fase *Planning*.	
Req ID	**Requirement**
1.2	Haruslah dapat mengurutkan data pelanggan (asc/des).
1.4	Haruslah dapat mengidentifikasi *trend* penjualan.
1.4.1	Haruslah dapat mengidentifikasi penurunan penjualan.

Tabel 2. Daftar *Restate Requirement* pada Fase *Planning*.

Setelah draft requirement di saring dengan melakukan *restate* terhadap *requirement,* aktivitas selanjutnya adalah melakukan pengelompokan kebutuhan-kebutuhan tersebut menjadi empat kategori yaitu apakah menjadi kebutuhan pemakai, kebutuhan sistem, kebutuhan operasi dan kebutuhan bisnis.

1. Kategori kebutuhan pemakai.

   Kebutuhan pemakai mendefinisikan aspek nonfungsional, bagaimana interaksi user dengan solusi. Yang termasuk dalam kebutuhan pemakai adalah antarmuka (tampilan aplikasi), performa aplikasi (tahan uji, ketersediaan, dan aksesbilitas), dan pelatihan (training) pemakai. Contoh kebutuhan pemakai diantaranya adalah :

   - Kasir haruslah dapat menangani lebih dari satu transaksi per menit.
   - Pelanggan haruslah dapat menyelesaikan pembelian sebuah produk melalui *website* dalam waktu lima menit.

2. Kategori kebutuhan sistem.

   Kebutuhan sistem mendefinisikan transaksi yang terjadi dan urut-urutanya didalam sistem, bagaimanan sistem yang baru berinteraksi dengan sistem yang sudah ada dan sebelum

membuat aplikasi yang baru tim proyek harus memahami infrastruktur yang ada diperusahaan agar akibat yang bersifat negatif dapat dikurangi. Contoh kebutuhan sistem diantaranya adalah :

- Semua aplikasi bisnis perusahaan mendukung notifikasi kepada pemakai secara *realtime* dan haruslah menerapkan komponen notifikasi yang telah sahkan.

3. Kebutuhan operasi.

   Kebutuhan operasi menggambarkan pengoperasian secara maksimal sebuah solusi dan meningkatkan layanan dengan mengurangi kemacetan dan resiko. Elemen-elemen kunci operasi diantaranya keamanan, ketersediaan, tahan uji, dapat dikelola, dapat skalabilitas, dan dukungan. Contoh kebutuhan sistem diantaranya adalah :

   - Ketersediaan : pelanggan haruslah dapat mengakses situs dan menggunakan sumberdaya pada setiap level layanan.
   - Skalabilitas : solusi haruslah dapat menangani berbagai volume pemakai dan transaksi.
   - Keamanan : data, layanan dan perangkat didalam sistem harus di lindungi dari akses yang tidak sah.
   - Dapat dikelola : website haruslah dapat di jalankan dan dikelola secara langsung maupun secara *remote* (melalui komputer lain).

4. Kebutuhan bisnis.

   Kebutuhan bisnis menggambarkan apa yang dibutuhkan perusahaan dan harapanya terhadap solusi. Contoh kebutuhan bisnis diantaranya adalah :

- Manager *call-center* haruslah dapat melihat informasi panggilan terakhir, panggilan saat ini, dan rata-rata jumlah panggilan untuk setiap operator telepon.
- Solusi harus didesain, dikompilasi dan dipasang secara cepat.
- Solusi haruslah dapat berinteraksi dengan proses binis, aplikasi dan sumberdata aplikasi lainya.

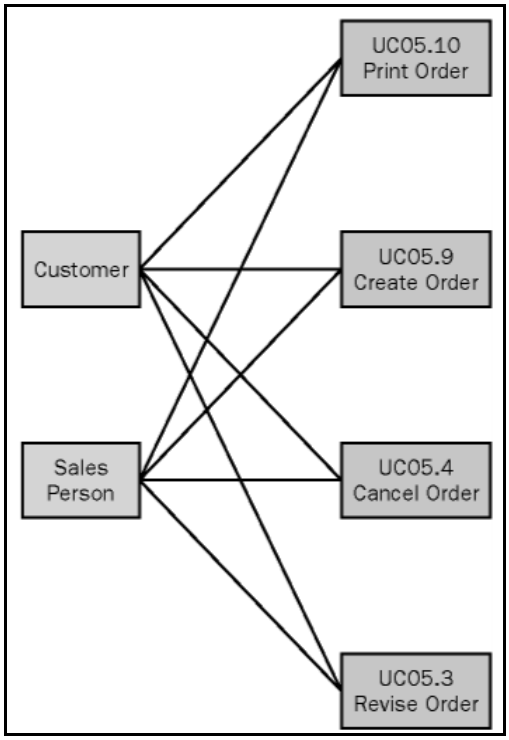
Pada tahap *envisioning* tim membuat diagram *usecase* level paling tinggi, dimana diagram *usecase* adalah daftar *usecase* yang ada didalam sistem dan digunakan untuk menetapkan jangkauan proyek. Selanjutnya adalah menyaring *usecase* pada tahap *planning* pada desain konsep ketika aktivitas *research*.

Aktivitas dalam menyaring usecase adalah sebagai berikut :

- Membuat subordinat usecase (membagi usecase menjadi beberapa usecase).
- Membuat *usage scenario* untuk setiap subordinat *usecase*.
- Memvalidasi setiap usecase dan *usage scenario* dari aktivitas *interview* yang sudah dilakukan, dokumen-dokumen lain dan dengan pemakai.
- Menyaring kebutuhan dengan usecase yang telah divalidasi dan informasi usage scenario.

Usecase menggambarkan bagian dari fungsionalitas sistem yang didalamnya terdapat *usage scenario* (sekenario bagaimana *usecase* tersebut difungsikan atau digunakan dengan melibatkan seorang aktor). Sebuah *usecase* dengan nama *Manage Order* dapat memiliki *subordinat usecase misalnya : Print Order, Create Order, Cancel Order* dan *Revise Order*. *Usecase* tersebut melibatkan dua aktor untuk menjalankan fungsionalitasnya yaitu aktor *Customer* dan *Sales*

Person. Aktivitas yang dilakukan untuk memecah *usecase* menjadi *subordinat-subordinat usecase* disebut dengan *refining usecase* (menyaring *usecase*).



Gambar 16. Menyaring Usecase.

Dibawah ini adalah contoh sebuah *high-level usage scenario* :

a. *Usage Scenario Usecase "Memesan Spesifikasi Produk"*.

Use case title	: Memesan Spesifikasi Produk
Abbreviated title	: Memesan Spesifikasi Produk
Use case ID	: UC 05.1
Requirements ID	: 15.1
Intent	: Untuk menyediakan spesifikasi produk lengkap kepada pelanggan.
Scenario narrative	: Pada saat pelanggan melihat item didalam katalog, pelanggan meminta spesifikasi-spesifikasi produk.

Use case title	: Memesan Spesifikasi Produk
Actors	: Pelanggan
Preconditions	: Pelanggan membuka katalog produk. Pelanggan melihat item didalam katalog .
Basic course	: 1. *Usecase* dimulai ketika pelanggan meng-klik Pemesanan Spesifikasi Produk untuk item yang dipilih. 2. Pelanggan memilih format dari spesifikasi produk. 3. Pelanggan memilih cara pengiriman produk. 4. Alamat pelanggan telah dikonfirmasi. 5. *Usecase* berakhir ketika pemesanan telah selesai, terkirim dan siap untuk menunggu kiriman katalog.
Alternative course	: -
Postconditions	: Pemesanan telah selesai, terkirim dan siap untuk menunggu kiriman katalog. Pelanggan kembali pada katalog yang dibuka terakhir kali.
Uses/extends	: Pelanggan membaharui profile, pelanggan membuat profile baru, dan mengirimkan pemesanan.
User implementation requests	: Tidak ada
Frequency	: Terjadi antara 17-100 sesi pelanggan di website.
Unresolved issues	: Tidak ada
Authority	: Mike Danseglio
Modification history **Author** **Description**	: 6 November 2006 : Heidi Steen : Versi Inisial

Tabel 3. *Usage Scenario* dari *Usecase*.

Usage scenario diatas menjelaskan bagaimana fungsi pemesanan spesifikasi produk yang dilakukan oleh konsumen melalui website. Jika dianalisa *useage scenario* tersebut masih sangatlah umum, sehingga dapat di buat lagi detail dari *usage scenario* misalnya "memesan spesifikasi produk melalui surat".

a1. *Subordinat Usage Scenario, Usecase "Memesan Spesifikasi Produk Melalui Surat ".*

Use case title	: Memesan Spesifikasi Produk Melalui Surat
Abbreviated title	: Memesan Spesifikasi Produk Melalui Surat
Use case ID	: UC 05.1.1
Requirements ID	: 15.1.1
Intent	: Untuk menyediakan spesifikasi produk yang dipilih kepada pelanggan melalui surat.
Scenario narrative	: Pelanggan memohon spesifikasi produk melalui layanan pengiriman surat. Pelanggan melihat-lihat katalog (UC 05.1)dan melihat item yang diminati, setelah memilih cara pengiriman melalui layanan pos surat, pelanggan mengisikan alamat pengiriman ...
Actors	: Pelanggan
Preconditions	: Pelanggan melihat-lihat katalog produk. Pelanggan melihat item produk di dalam katalog.
Basic Course : 1. Usecase dimulai ketika meng-klik Pemesanan Spesifikasi Produk untuk item yang dipilih. 2. Pelanggan memilih format berupa lembar spesifikasi atau brosur. 3. Pelanggan memilih cara pengiriman apakah melalui layanan pos elektronik, kantor pos atau pengiriman dalam satu hari. 4. Daftar alamat dari profile ditampilkan. 5. Pelanggan memilih alamat dari daftar alamat-alamat. 6. Alamat telah dikonfirmasi. 7. Pelanggan mengirimkan permintaan. 8. *Usecase* berakhir pada saat pelanggan memberikan nomor konfirmasi.	
Alternative course : 1. Jalan alternatif dimulai pada langkah no. 4 2. 4a. Alamat tidak ada pada profile. 4b. Menuju ke form membuat profile baru atau membaharui profile yang ada. 3. *Usecase* dimulai lagi pada langkah no.5	
Postconditions	: Pemesanan telah selesai, terkirim, Pelanggan kembali pada katalog yang dibuka terakhir kali.
Uses/extends	: Perluasan UC 05.1, Pemesanan Spesifikasi Produk
User implementation requests : Tidak ada	

Use case title	: Memesan Spesifikasi Produk Melalui Surat
Frequency	: Terjadi antara 17-100 sesi pelanggan di website.
Unresolved issues	: Tidak ada
Authority	: Mike Danseglio
Modification history **Author** **Description**	: 6 Desember, 2006 : Heidi Steen : Versi Inisial

Tabel 4. *Usage Scenario* dari *Subordinat Usecase*.

5.1.2. Desain Logik.

Aktivitas tim pada desain logikal adalah memecah permasalahan yang muncul menjadi unit-unit kecil yang disebut dengan modul. Modul merupakan unit logikal yang digunakan untuk membuat abstraksi usecase dan skenario yang dibuat pada desain konseptual. Untuk setiap modul yang dihasilkan, tim akan mengidentifikasi *objects*, *services* dan *attributes* dan *realationship*. Tim juga mengidentifikasi kandidat teknologi untuk solusi yang akan dirancang selama desain logikal. Pada desain logik ini peran tim akan di bagi sebagai berikut :

Peran Tim	Tugas Utama	Tugas Kedua
Product management	Memastikan bahwa desain sudah sesuai dengan kebutuhan bisnis dan pemakai.	Mengakomodir harapan-harapan pemakai.
Program management	Bertanggung jawab untuk deliverables desain logik	Mendevinisikan rencana proyek yang meliputi : resources, schedules, and risk assesment
Development	Mengidentifikasi layanan dan objek-objek yang berhubungan	Bertindak sebagai konsultan teknologi untuk mengevaluasi prototype teknologi
Testing	Memvalidasi model desain logik	Mendefinisikan rencana pengujian level tinggi
User	Mendefinisikan performa	Mendefinisikan rencana

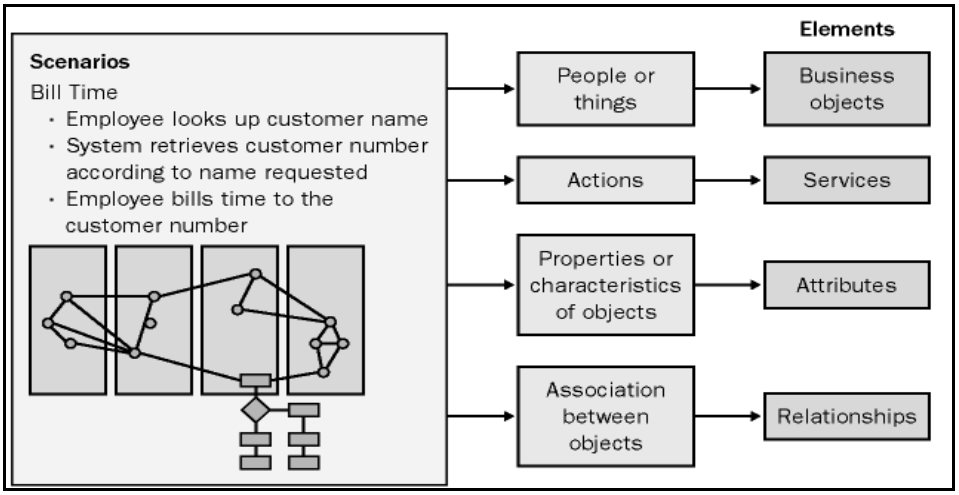
Peran Tim	Tugas Utama	Tugas Kedua
experience	user dan solusi yang direkomendasikan	edukasi bagi user
Release management	Mengevaluasi feasibility penerapan solusi	Mendefinisikan infrastruktur dan pemasangan solusi

Tim akan melakukan beberapa pertimbangan diantaranya adalah pertimbangan bisnis, arsitektur enterprise, dan teknologi yang dipilih. Pertimbangan bisnis akan sangat berhubungan erat dengan :

- *Feasibility* : menetapkan apakah teknologi sesuai dengan kebutuhan bisnis dan sesuai dengan *requirement*.
- *Product Cost* : biaya produksi, termasuk didalamnya adalah *developer*, *server*, *reseller license* dan biaya *upgrade.* Tim juga mempertimbangkan kebutuhan awal *hardware* dan *software*, *support, infrastruktur,* dan *training*.
- *Experience* : tenaga ahli yang dimiliki untuk kebutuhan *training,* konsultasi dan tingkat kenyamanan.
- *Return Of Investment* : investasi harus memiliki keterkaitan dengan pengembalian investasi.
- *Maturity* : produk yang dapat diterima oleh masyarakat, stabil, mudah digunakan dan adanya dukungan sumberdaya.
- *Supportability* : pemilihan teknologi yang sesuai dan mendukung pengembangan solusi jangka panjang.

Tim juga harus mempertimbangkan teknologi yang dipilih, beberapa pertimbangan teknologi adalah masalah yang berhubungan dengan keamanan, standar layanan interaksi, akses data, penyimpanan data, layanan sistem, perangkat pengembangan, dan

operating sistem. Selanjutnya melakukan identifikasi terhadap objek bisnis. Objek dapat berupa orang atau berbagai hal yang dideskripsikan didalam *usage scenario*. Objek-objek bisnis merupakan dasar dari *services, attributes* dan *relations* .



Gambar 17. Analisa Desain Logikal.

Bagaimana mengidentifikasi objek bisnis, layanan (service) dan attribut ?

1) Objek bisnis dapat ditemukan didalam diagram *usecase* dan *usage scenario* yang dapat berupa orang atau apapun, misalnya : pekerja, pelanggan, kontrak, nomor pelanggan.
2) Layanan (*service*) adalah aksi yang dilakukan oleh objek bisnis yang dapat ditemukan didalam *usage scenario*, sebagai contoh menghitung total pembelian dan menetapkan biaya pengiriman.
3) Atribut atau properti suatu objek adalah sebuah nilai data yang melekat pada objek. Sebagai contoh objek bisnis yang bernama *Customer* memiliki atribut No_Akun, Nama, Alamat dan No_telepon.

4) Hubungan (*relationship*) menggambarkan bagaimana sebuah objek terhubung dengan objek lain. UML mendefinisikan relationship empat macam relationship, yaitu :

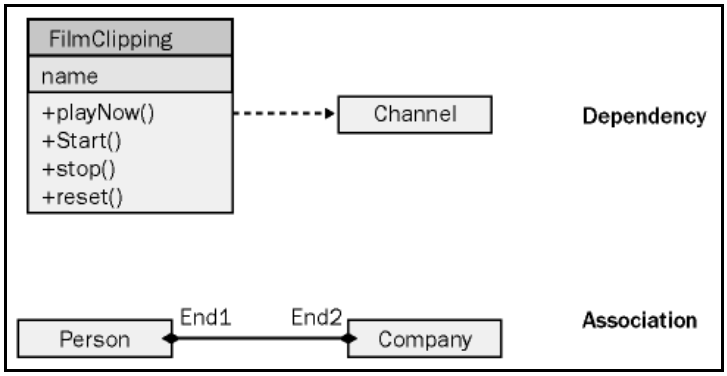
a) *Dependency*, relasi antara dua objek dimana melakukan perubahan pada sebuah objek akan berpengaruh pada perilaku dan layanan objek lain. Relasi *dependency* digambarkan dengan garis yang terputus.

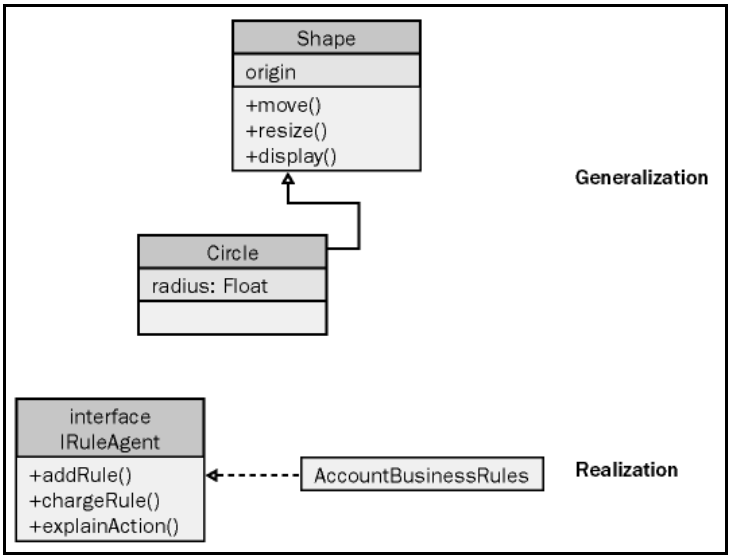
b) Association, hubungan yang menggambarkan struktur antara dua objek. *Agregation* merupakan tipe khusus dari relasi *association* yang merepresentasikan hubungan antara sebuah objek yang utuh dengan bagian-bagianya. *Composition* adalah tipe relasi agregasi dimana objek yang utuh mengatur *lifetime* (masa hidup) bagian-bagianya. Relasi *association* digambarkan dengan garis yang solid.

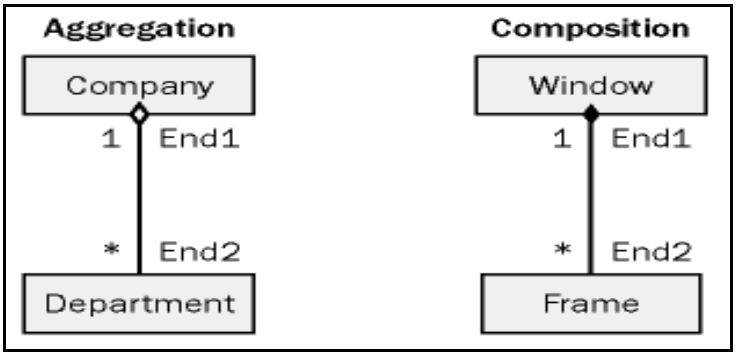
c) Generalization, relasi antara sesuatu yang lebih umum (disebut *parent*) dengan sesuatu yang lebih spesifik (disebut *child*). Sebagai contoh manager adalah tipe yang lebih spesifik dari karyawan. Objek *Child* akan memperoleh atribut atau properti dari objek *parent*-nya. Relasi *generalization* digambarkan dengan garis yang solid dengan anak panah dari objek *child* ke objek *parent*.

d) Realization, merupakan relasi antar kelas (*class*) dimana sebuah abstrak kelas menetapkan kontrak dengan kelas lain dimana kelas tersebut perlu untuk memuatnya. Hubungan realization terjadi antara antarmuka (tampilan) dan kelas-kelas dan komponen yang merealisasikan antarmuka.

Dibawah ini digambarkan contoh relasi antara kelas-kelas dalam perancangan solusi pada desain konseptual. Perangkat lunak yang dapat digunakan untuk penggambaran model yaitu Rational Rose dan Microsoft Visio.







Gambar 19. Tipe Relasi Antar Kelas.

Selama desain konseptual tim mulai mengidentifikasi teknologi yang mungkin dapat digunakan sebagai bagian dari solusi. Maka pada tahap desain logik tim akan menyaring kandidat teknologi yang telah diidentifikasi. Dalam mengevaluasi kandidat teknologi haruslah mempertimbangkan teknologi yang dipilih, bisnis yang sedang berjalan, dan arsitektur perusahaan.

5.1.3. Desain Fisikal.

Desain fisikal akan di lakukan setelah semua anggota tim setuju bahwa mereka memiliki informasi yang cukup dari desain logikal yang telah diselesaikan. Tim pengembang akan menerapkan teknologi yang telah dipertimbangkan sebelumnya dengan batasan-batasan pada konseptual dan logikal desain. Tim akan memulai mendiskripsikan komponen, layanan, dan teknologi dari perspektif kebutuhan pengembangan. Fisikal desain mendifinisikan bagian dari solusi yang akan dikembangkan dan bagaimana solusi harus dikembangan.

Desain fisikal bukan aktivitas pemrograman (*coding*) atau penyebaran teknologi, melainkan merancang spesifikasi komponen

secara detil untuk pengembangan, menetapkan dimana komponen akan ditempatkan dan mengidentifikasi teknologi yang dapat digunakan untuk pengembangan solusi. Aktivitas pada desain fisikal diantaranya adalah :

- Mengidentifikasi teknologi yang sesuai untuk pengembangan dengan melakukan evaluasi terhadap teknologi yang terbaik.
- Mentransformasikan desain logikal kedalam desain fisikal.
- Menyedikan *baseline* untuk proses pengembangan, sebagai tambahan pada fase ini adalah dengan pembuatan model dan strategi, mendefinisikan aturan dalam proses pengembangan dan tanggung jawab personel pada peran masing-masing.
- Mendefinisikan bilamana rencana proyek disetujui dan *milestone* telah tercapai.

Pada akhir fase desain fisikal, anggota tim proyek memiliki dokumentasi yang cukup lengkap untuk mulai membangun solusi. Beberapa yang dapat dihasilkan pada fase ini adalah :

- Diagram-diagram kelas dari solusi.
- Model komponen, sequence diagram, dan diagram aktivitas.
- Skema database dari solusi.
- *Baseline* model penyebaran yang menyediakan topologi jaringan, lokasi dan interkoneksi *hardware*, topologi data dan komponen, lokasi komponen, layanan dan penyimpanan data .
- Spesifikasi komponen yang terdiri dari struktur dan antarmuka.
- Strategi pemaketan dan distribusi yang mengidentifikasi layanan yang akan dipaketkan bersama dan bagaimana komponen didistribusi melalui topologi jaringan.
- Model pemrograman dan dokumentasi kode.

Dengan menerapkan tiga tahapan desain pada fase planning maka kebutuhan-kebutuhan perancangan perangkat lunak telah didefinisikan dengan lengkap dan jelas. Tim pengembang dapat memulai fase-fase berikutnya sampai solusi dapat distribusikan pada lokasi yang telah direncanakan. Tahapan-tahapan pada desain fisikal adalah sebagai berikut :

1. *Research*, pada tahap ini tim proyek menetapkan batasan dan kebutuhan sistem secara fisikal dan mengidentifikasi perubahan terhadap infrastruktur.

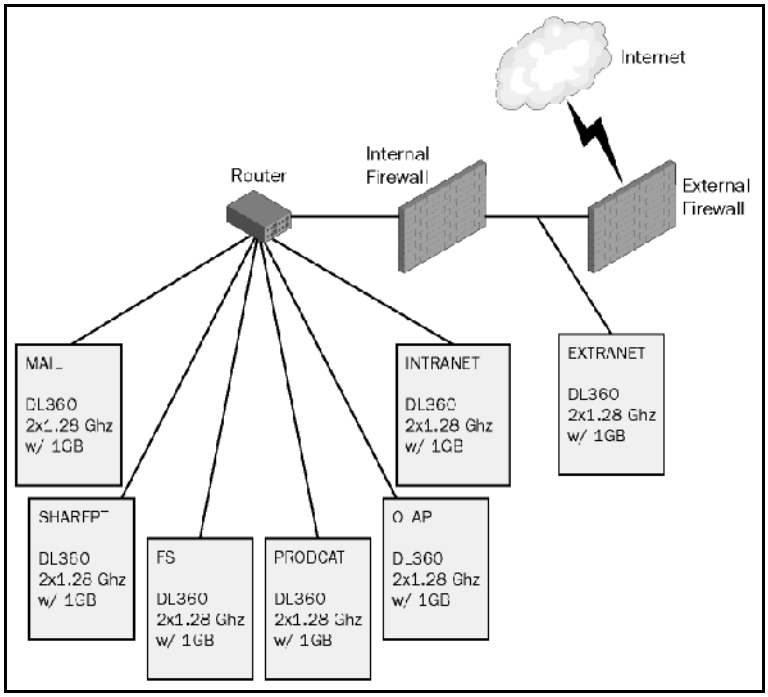
   Tim mengumpulkan kebutuhan fisikal seperti unjuk kerja, biaya dan keuntungan, kemudahan penggunaan, dapat disebarkan, dapat didukung, reliabel, dan penggunaan ulang. Sedangkan batasan fisikal adalah anggaran, penjadualan, topologi jaringan, topologi data, topologi komponen, petunjuk teknologi, dan keamanan. Apabila terjadi konflik antara kebutuhan dan batasan maka tim harus menetapkan beberapa *tradeoff* matriks sebelum pengembangan dimulai, identifikasi area infrastruktur yang menyebabkan konflik, analisa gap antara kebutuhan dan batasan kemudian pecahkan permasalahanya, terakhir dengan melakukan tukar pendapat dengan semua yang terlibat dalam proyek.
2. *Analysis,* pada tahap ini tim proyek melakukan pengembagan awal model penyebaran dan memilih teknologi yang akan digunakan. Tim melakukan penyaringan terhadap model UML yang dibuat pada desain logikal yang meliputi objek dan layanan, diagram *class*, diagram *sequence*, diagram *activity*, dan diagram *component*.
   a. Objek dan layanan dapat dikelompokkan menjadi layanan terhadap pemakai, terhadap bisnis, terhadap data, terhadap

sistem dan mengidentifikasi layanan-layanan yang mungkin tersembunyi.

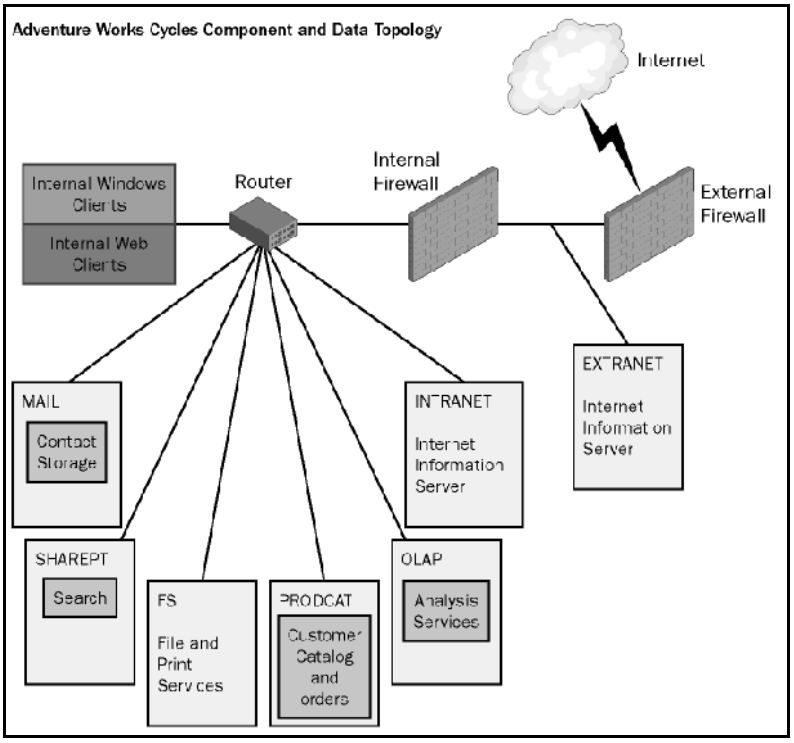
b. Diagram *class* untuk menggambarkan model objek aplikasi, pada tahap ini tim menyaring diagram *class* yang meliputi :
   - Mentransformasi objek logikal kedalam definisi *class* termasuk antarmukanya.
   - Mengidentifikasi objek yang tidak jelas pada saat desain logik.
   - Malakukan penggabungan objek logikal jika diperlukan.
   - Mengelompokkan objek kedalam model layanan yaitu objek *boundary* (layanan pemakai), objek *control* (layanan bisnis), dan objek *entity* (layanan data).
   - Menyaring methoda dengan fokus pada parameter, penggunaan yang berlebih, kombinasi atau pembagian methoda, dan penanganan dalam melewatkan nilai.
   - Menyaring atribut dengan fokus pada atribut yang terlindungi yang akan digunakan oleh objek turunanya.
c. Diagram *sequence* yang menggambarkan interaksi antar objek, pada tahap desain fisikal tim melakukan penyaringan diagram sequence untuk memasukkan interasi antara *class* atau layanan pada batasan fisikal atau kebutuhan teknologi.
d. Diagram *activity* yang menggambarkan transisi dan aliran aplikasi, pada tahap ini menyertakan platform fisikal dan kebutuhan teknologi, serta identifikasi aliran proses yang potensial.

e. Diagram *component* yang menggambarkan ketergantungan atar komponen, pada tahap ini diklarifikasi ketergantungan antar komponen dan lebih lanjut mendefinisikan pemaketan.

f. Model penyebaran solusi yang meliputi topologi jaringan yang menggambarkan pemetaan lokasi dan interkoneksi antar perangkat keras seperti komputer *server*, komputer *client* dan perangkat keras lainya. Sedangan topologi data dan komponen menggambarkan pemetaan lokasi paket, komponen dan layananya yang berhubungan dengan topologi jaringan dan lokasi penyimpanan data. Berikut ini contoh dari topologi jaringan dan topologi komponen dan data dari studi kasus pada perusahaan Adventure Work Cycle.



Gambar 20. Topologi Jaringan.



Gambar 21. Topologi Data dan Komponen.

3. *Rationalization*, tim menetapkan strategi pemaketan aplikasi dan penyebaran, memaketkan komponen dan layanan, serta mendistribusikan komponen pada topologi jaringan. Tujuan dari rationalization adalah mendistribusikan layanan dan paket kedalam komponen (file aplikasi berupa .dll atau .exe) Sedangkan strategi distribusi adalah dasar pemikiran untuk menetapkan dimana sebuah layanan ditempatkan pada arsitektur solusi. Maka strategi pemaketan adalah dasar pemikiran untuk menetapkan dimana layanan-layanan berada didalam setiap komponen. Dalam menetapkan strategi pemaketan haruslah mempertimbangkan :

   a. Manajemen status.

   Manajemen status adalah proses pengelolaan status dan informasi halaman pada saat terjadi banyak permintaan

(*request*) halaman yang sama atau halaman yang berbeda dari pemakai. Beberapa metoda dalam mengelola status pada aplikasi berbasis web adalah :

- Menyimpan status di komputer *client* untuk mempertahankan sebuah nilai ketika terjadi permintaan yang sama oleh pemakai.
- Penggunaan *structured query language* (SQL) untuk melakukan *query*.
- *Cookies*, merupakan data teks yang berukuran kecil yang tersimpan dikomputer *client* atau di memori pada *session browser client*.
- *Application state*, merupakan mekanisme penyimpanan data yang dapat diakses pada semua halaman web.
- *Session state,* merupakan mekanisme penyimpanan data yang dapat diakses hanya dari sebuah sesi halaman web aktif (*current session)*.

b. Desain aplikasi.

- Skalabilitas : kecepatan dan kemudahan penggunaan solusi untuk menangani bnayak transaksi dan banyak pemakai.
- Performa : waktu tanggap sistem dan kecepatan sistem menjalankan aplikasi.
- Pengelolaan : sistem dapat dikelola pada setiap tingkatan (level).
- Reuse : komponen-komponen sistem dapat digunakan oleh aplikasi lainya.
- Konteks bisnis : pembagian fungsi-fungsi bisnis.

- Granularitas : sejumlah layanan dan objek yang dipaketkan didalam komponen tunggal.

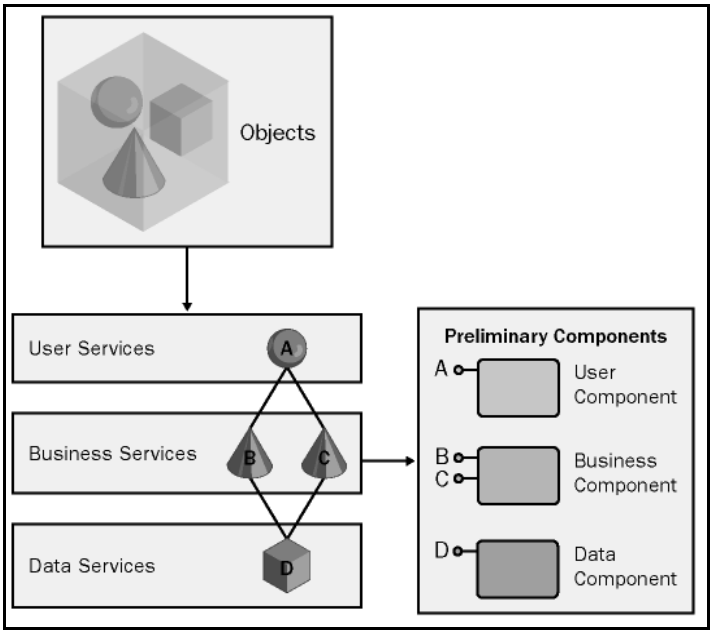
Rencana perancangan komponen yang bagus didasarkan pada *high cohession* dan *loose coupling*. *Cohesion* adalah hubungan antara elemen internal yang berbeda didalam sebuah komponen. Sementara *Coupling* adalah hubungan antara komponen dengan komponen lain. *Cohesion* dapat berupa :

- Fungsional : sebuah unit yang menjalankan satu tugas (*strongest cohesion*).
- Sekuensial : sebuah unit yang mengandung operasi yang harus dijalankan untuk tugas tertentu dan harus membagi data yang sama.
- Komunikasional : operasi didalam unit menggunakan data yang sama tetapi tidak saling berhubungan.
- Temporal : operasi-operasi dikombinasikan karena dijalankan secara simultan.

Coupling dapat memiliki kondisi yang ketat (*tightly coupled*) atau bebas (*loosely coupled*), ketika sebuah komponen berkondisi ketat maka komponen bergantung pada komponen eksternal untuk menyelesaikan sebuah fungsi. Jika komponen berkondisi bebas maka komponen tidak bergantung pada eksternal komponen ketika menjalankan sebuah fungsi.

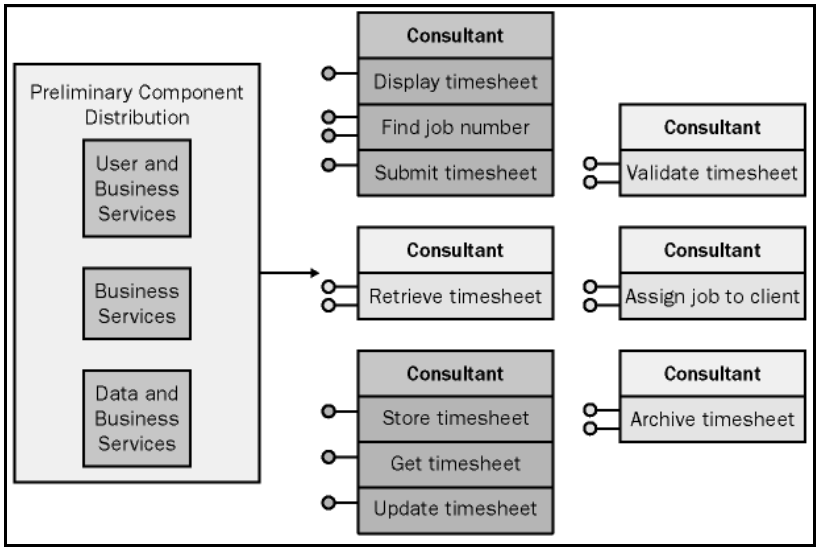
Untuk memulai proses distribusi paket layanan, maka harus diidentifikasi layanan-layanan utama didalam model objek dan memecah objek tersebut menjadi layanan-layanan berbasis lapisan. Dalam setiap objek bisnis memiliki layanan-layanan yang terbagi menjadi tiga lapisan yaitu : layanan pemakai, layanan

bisnis dan layanan data. Masing-masing lapisan digambarkan berbeda satu sama lain. Layanan pemakai digambarkan dalam bentuk bulat, sedangkan layanan bisnsi digambarkan dalam bentuk kerucut, dan layanan data digambarkan dalam bentuk kotak. Sebuah objek yang dipaketkan dalam komponen dibungkus menjadi satu seperti pada gambar berikut.



Gambar 22. Tiga lapisan layanan dalam komponen.

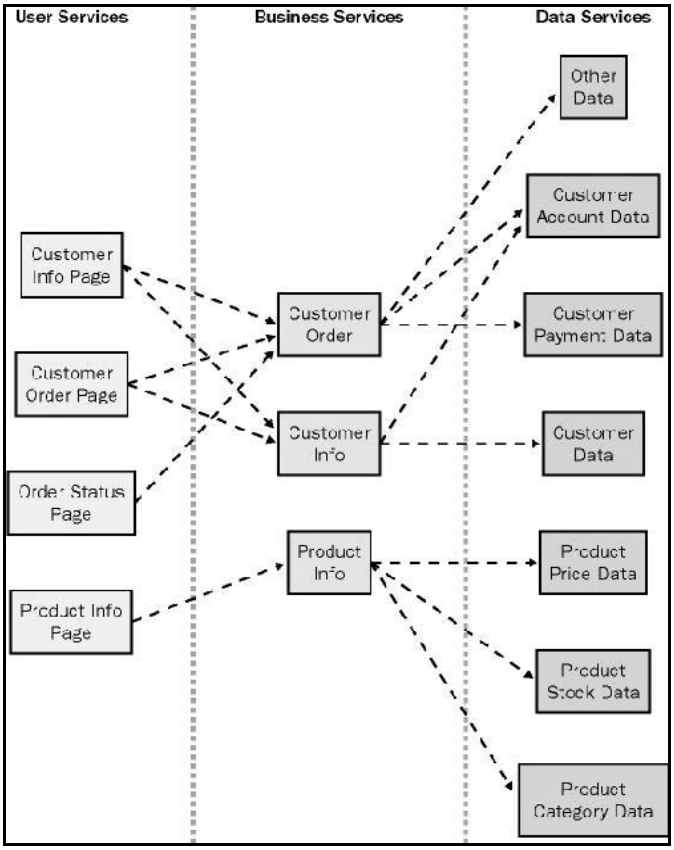
Setelah memaketkan layanan dalam suatu komponen selanjutnya didistribusikan melalui topologi jaringan dan merancang topologi komponen.



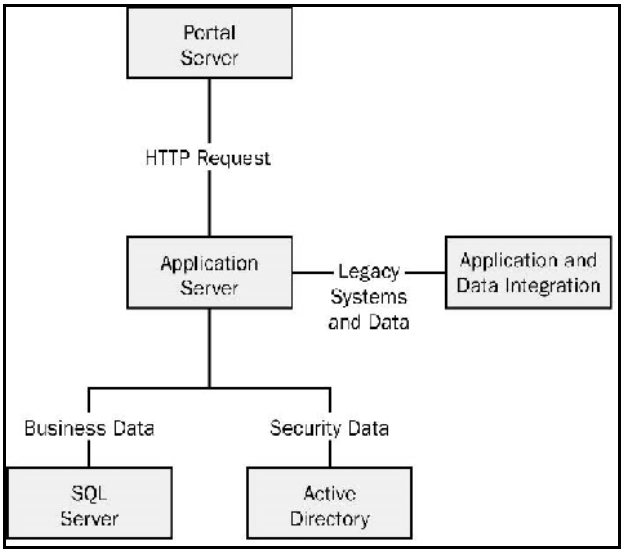
Gambar 23. Persiapan distribusi komponen.

Untuk membantu pendistribusian lapisan ikutilah petunjuk berikut :

a) Distribusikan layanan pemakai di webserver atau di komputer klient.
b) Disitribusikan layanan bisnis di aplikasi server atau webserver.
c) Distribusi layanan data di database server atau lokasi dimana data akan disimpan pada topologi jaringan.
d) Distribusikan komponen sesaui lapisan yang telah ditetapkan.



Gambar 24. Contoh model komponen.



Gambar 25. Contoh model penyebaran.

4. *Implementation*, tim menetapkan model pemrograman, membuat spesifikasi antarmuka komponen, atribut dan layanan. Model pemrograman menjelaskan bagaimana komponen-komponen akan dibuat struktur dalam kode program. Beberapa pertimbangan dalam model pemrograman diantaranya adalah :
   a) Implementasi teknologi, meliputi bahasa pemrograman, antarmuka pemrograman aplikasi (API), server dan teknologi server, serta teknologi-teknologi lainya.
   b) Objek berstatus atau tanpa status, objek berstatus mempertahankan informasi yang digunakan dalam satu atau lebih pemanggilan fungsi oleh *client*. Sedangkan objek tanpa status tidak dapat mengelola informasi dalam suatu transaksi.
   c) Pemanggilan fungsi didalam proses atau diluar proses, pemanggilan didalam proses : komponen melaksanakan eksekusi didalam proses tunggal (akan meningkatkan performa). Pemanggilan diluar proses : komponen melaksanakan tugasnya dengan membagi proses berdasarkan pemakai.
   d) Kohesi dan kopling, *Cohesion* merupakan hubungan antara elemen internal yang berbeda didalam sebuah komponen. Sementara *Coupling* adalah hubungan antara komponen dengan komponen lainya.
   e) Mode terhubung dengan mode tak terhubung, pada lingkungan komponen yang terdistribusi, berbagai komponen berpartisipasi didalam sebuah layanan secara *real-time* dengan mode yang selalau terhubung, apabila interaksi gagal (hubungan terputus) maka komponen dapat dijalankan pada

mode tak terhubung dan dapat menghubungkan kembali jika dibutuhkan.

f) Sinkronisasi dan non sinkronisasi, model pemrograman sinkronisasi yaitu pemanggilan komponen dari melanjutkan dengan proses lain sampai antarmuka yang dipanggil selesai memberikan layanan yang diminta dan pengembalian kontrol ke komponen yang dipanggil. Sedangkan pemrograman non sinkronisasi mengijinkan komponen mengirimkan pesan ke komponen lain dan melanjutkan fungsi tanpa harus menunggu balasan. Pemrograman non sinkronisasi lebih kompleks, namun dengan pemrograman COM+ menjadi lebih mudah.
g) Model *threading*, model ini sangat rumit karena model yang tepat bergantung pada fungsi didalam komponen.
h) Penanganan kesalahan (*error handling*), oleh karena komponen dapat menjalankan fungsinya secara sempurna maka diperlukan sebuah penanganan kesalahan jika terjadi kesalahan ketika menjalankan fungsi.
i) Keamanan, mencakup kemanan komponen, kemanan database, konteks interaksi kepada pemakai dan kemanan berbasiskan peran dalam *group*.
j) Distribusi, pada distribusi aplikasi tiga layer logik tidaklah selalu di terjemahkan menjadi tiga layer distribusi sescara fisik. Misalnya layer layanan pemakai dan layanan bisnis diletakkan diserver dan layer data diletakkan di server yang berbeda.

Setelah mendiskripsikan model pemrograman, selanjutnya tim mendefinisikan bagaimana komponen akan berinteraksi.

Interaksi ini akan didokumentasikan oleh antarmuka komponen dengan mendiskripsikan bagaimana mengakses layanan dan atributnya.

Selanjutnya tim merancang layer presentasi yang menyediakan interaksi pemakai dengan sistem. Pemakai dapat dibedakan menjadi pemakai orang dan pemakai komputer pada sistem lain. Terakhir pada tahap desain fisik tim basisdata harus mempertimbangkan batasan data fisik sperti memori dan ukuranya, tuning performa, kunci primer dan kunci tamu, triger, *stored procedure*, model objek aplikasi, spesifikasi *indexing*, partisi data, migrasi data dari sistem sebelumnya dan pertimbangan operasional termasuk proses update dan backup.

BAGIAN 6

MENDESAIN LAPISAN PRESENTASI

6.1. Dasar Mendesain Antarmuka Pemakai

Lapisan presentasi adalah bagian dari aplikasi bisnis yang menyediakan mekanisme komunikasi antar pemakai dan lapisan layanan bisnis dari sistem. Komponen antarmuka pemakai mengatur interaksi dengan pemakai dengan menampilkan data, meminta input data dari pemakai, menterjemahkan *even* yang dilakukan oleh pemakai, merubah status antarmuka pemakai, dan membantu pemakai melihat kemajuan tugas mereka. Komponen antarmuka pemakai memiliki fungsi yang meilputi :

1. Memperoleh masukan data dari pemakai dan membantu pemakai dengan menampilkan *tooltips*, validasi dan kontrol yang sesuai.
2. Menangkap kejadian dari pemakai dan memanggil fungsi pengontrol untuk memberitahu komponen antarmuka merubah tampilan data .
3. Menolak data yang dapat diisikan pemakai karena tidak sesuai tipedata.
4. Melaksanakan validasi data yang dimasukkan, misalnya jangkauan data.
5. Memformat data yang dimasukkan misalnya *date*, *e-mail* dan telepon.

Mendesain antarmuka pemakai haruslah mengimplementasi-kan tugas-tugas pemakai dengan cara meyesuaikan dengan intuisi pemakai. Berhasil atau tidaknya aplikasi tergantung dari antarmuka, ketika pemakai merasa kesulitan menggunakan aplikasi maka aplikasi dapat dikatakan tidak berhasil. Pertanyaan-pertanyaan yang harus dijawab ketika mendesain antarmuka pemakai adalah :

1. Bagaimana pemakai berinteraksi dengan aplikasi ?
2. Apakah antarmuka merepresentasi konsep dan terminologi pemakai ?
3. Kiasan-kiasan apa yang sesuai digunakan dalam mendesain antarmuka ?
4. Apakah pemakai dapat dengan mudah menemukan fitur yang dibutuhkan untuk menyelesaikan tugas-tugas yang utama ?
5. Apakah aliran kerja benar dan lengkap ?
6. Apakah pemakai mudah dalam mengakses bantuan ketika muncul masalah ?

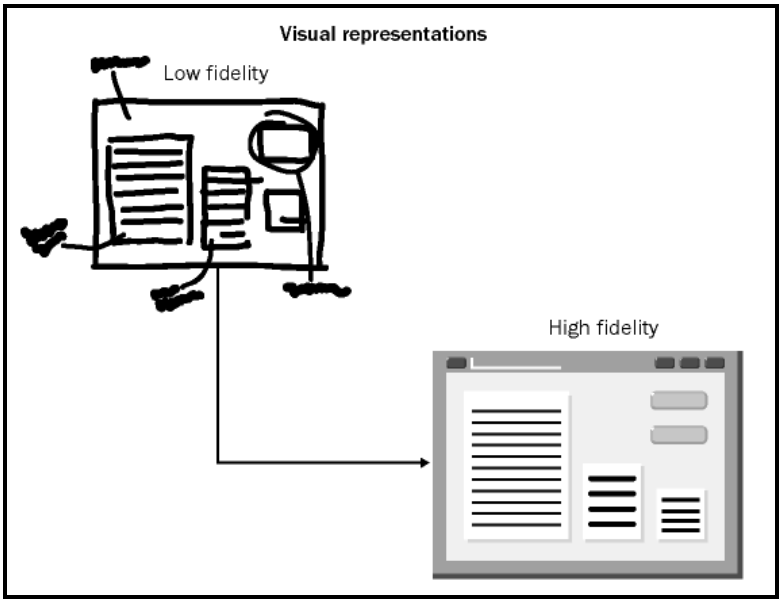
Berikut ini adalah fitur-fitur yang dapat membuat antarmuka menjadi efektif :

1. Intuituf desain : desain antarmuka dimana pemakai dapat menggunakan intuisinya dalam memahami bagaimana menggunakan aplikasi.
2. Optimal area layar: menetapkan konten antarmuka dengan merencana- kan sejumlah informasi yang ditampilkan dan sejumlah input yang dimasukkan oleh pemakai.
3. Tampilan : dengan mempertimbangkan faktor-faktor seperti frekwensi dan waktu pemakai berinteraksi dengan bagian spesifik dari antarmuka untuk menentukan tampilan antarmuka.

4. Kemudahan navigasi : karena berbeda pemakai akan berbeda cara dalam mengakses komponen pada antarmuka maka desain komponen haruslah mudah diakses menggunakan tombol tabulasi, panah, tombol jalan pintas (*shortcut*) dan penggunaan mouse.
5. Navigasi yang terkontrol, penting untuk menetapkan urutan komponen yang diakses oleh pemakai.
6. Populasi nilai standar, jika antarmuka mengandung *field* standar, maka akan lebih baik menyediakan nilai yang otomatis.
7. Validasi masukkan, penting untuk memvalidasi masukan sesuai dengan tipedata atau format tertentu sebelum diproses.
8. Menu, *toolbar* dan *help*, desain antarmuka haruslah memiliki menu, *toolbar* dan *help* untuk kemudahan penggunaan aplikasi oleh pemakai.
9. Efisiensi penanganan *event*, agar pemakai tidak menunggu lama respon dari aplikasi.

6.2. Desain Awal Antarmuka Pemakai

Bagian pertama dalam mendesain antarmuka pemakai adalah membuat desain awal yang akan di tinjau ulang oleh pemakai. Desain awal dapat dilakukan dengan membuat gambar di kertas (*low fidelity*) atau menggunakn *tool* peracangan antarmuka misalnya lingkungan pengembangan *visual basic*. Gambar berikut ini menunjukkan perbedaan antara *low fidelity* dan *high fidelity*.



Gambar 26. Desain Layer Presentasi.

Pada rancangan antarmuka kita dapat membuat peta navigasi yang akan digunakan untuk pemanggilan komponen melalui *event-control*, selain itu pertimbangkan juga validasi terhadap *field* dan pemrosesan kesalahan yang mungkin terjadi. Antarmuka sebaiknya menyediakan *user-assistance* untuk membantu pemakai dalam menggunakan aplikasi. *User assistance* meliputi :

1. *Online Help* (bantuan online), merujuk pada layanan bantuan yang tersedia selama pemakai menggunakan aplikasi. *User-assistance* dapat dipasang bersama dengan aplikasi, di alokasikan didalam CD, atau di alokasikan melalui jaringan intranet atau internet.
2. *Tooltips* (perangkat tip), berbentuk label kecil yang ditampilkan ketika pemakai memindahkan kursor diatas kontrol atau bagian-bagian didalam antarmuka aplikasi.

3. *Status Display* (tampilan status), menyediakan instruksi atau pesan yang ditampilkan didalam kontrol statusbar atau label.
4. *Wizard* (tahapan), menyediakan panduan bagi pemakai berupa prosedur atau tahapan-tahapan. Biasanya digunakan untuk membantu tugas-tugas yang kompleks dan membutuhkan waktu yang cukup lama untuk menyelesaikanya.
5. *Accessbility Aids* (bantuan pengaksesan), adalah program khusus yang membantu pemakai agar lebih efektif dalam menggunakan aplikasi. Bantuan tersebut dapat berupa :
   - Pembesaran layar antarmuka.
   - Pembacaan layar antarmuka teks atau gambar dalam bentuk suara.
   - Penterjemahan suara yang akan menjadi input bagi aplikasi.

Dalam perancangan aplikasi sangatlah penting untuk menetapkan pilihan model antarmuka karena akan berpengaruh pada penyebaran, bagaimana pemakai berinteraksi dan menggunakan data dan mengelolaan status selama aplikasi digunakan oleh pemakai. Model antarmuka aplikasi dapat berupa :

1. Antarmuka aplikasi yang berbasiskan *window* (jendela).
   Model ini digunakan untuk perancangan aplikasi *offline* dan sangat kaya dengan interaksi pemakai biasanya pada komputer *standalone* (berdiri sendiri). Ada tiga kategori antarmuka *standalone* :
   - Fitur penuh untuk komputer kerja (*workstation*), antarmuka aplikasi berupa *form window* dan kontrol-kontrol.
   - Penyisipan tag HTML, dengan menambahkan HTML didalam aplikasi berbasiskan windows dengan fleksibilitas *runtime*

yang cukup baik karena HTML dapat dipanggil dari sumber daya eksternal.

- Aplikasi tambahan, yaitu aplikasi-aplikasi lain yang ditambahkan kedalam solusi seperti *add-in* microsoft office, Autocad, CRM, dan aplikasi lainya.
- Akses secara *remote*, dengan menggunakan aplikasi *remote* seperti windows xp remote desktop dan terminal server sehingga *client* dapat menjalankan sistem operasi dan aplikasi secara *remote* (dari jarak jauh).

2. Antarmuka aplikasi yang berbasiskan *web (*penggunaan *browser*)*.* Model ini digunakan untuk perancangan aplikasi *online* dimana aplikasi dapat diakses secara konkuren dan simultan. Dengan bahasa ASP.NET misalnya disediakan antarmuka yang kaya fitur yang mencakup :
   - Lingkungan pengembangan yang konsisten.
   - Pengikatan data pada antarmuka web.
   - Antarmuka berbasis kontrol server maupun HTML.
   - Terintegrasi dengan model keamanan .NET.
   - Pilihan manajemen status dan *chacing.*
   - Ketersediaan, performa, skalabilitas pemrosesan web.
3. Antarmuka aplikasi yang berbasiskan *mobile device* (perangkat bergerak)*.*
   - Antarmuka untuk aplikasi perangkat bergerak haruslah memiliki kemampuan menampilkan informasi dengan ukuran layar yang lebih kecil dibandingkan aplikasi pada umumnya. Termasuk dalam perangkat bergerak adalah komputer genggam, telepon dengan teknologi WAP, atau PDA.

- Oleh karena interaksi pemakai dengan aplikasi terasa lebih sulit maka desain antarmuka dibuat sangat minimal dengan kebutuhan input data yang sedikit.

4. Antarmuka aplikasi yang berbasiskan dokumen.

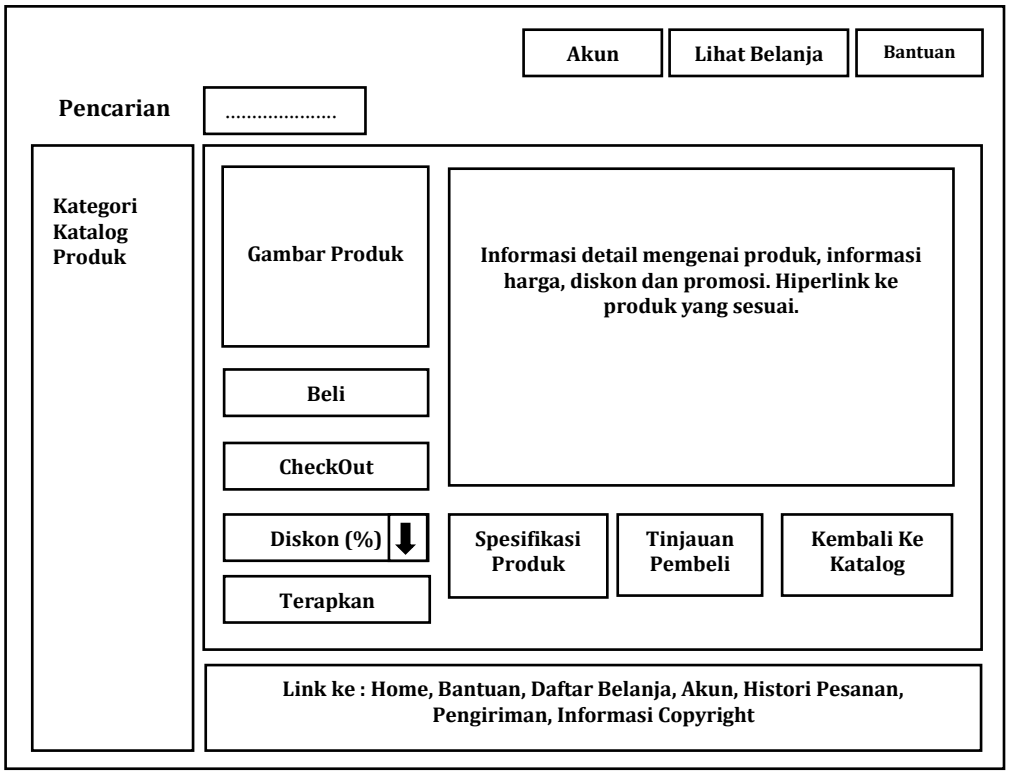
   Beberapa aplikasi mungkin menggunakan *form* dalam bentuk dokumen untuk kebutuhan :

   - Memasukkan dan menampilkan data misalnya dengan aplikasi *microsoft office* seperti *word,* laporan penjualan dengan lembar kerja excel atau untuk pengelolaan proyek dengan *microsoft project.*
   - Memperoleh data, pemakai melengkapi pengisian formulir, akuntan memasukkan data melalui lembar kerja dan perancang sistem membuat desain dengan *microsoft visio*.

Dalam mendesain antarmuka aplikasi, perhatikan bagian dari *usecase* yaitu *precondition, basic course, alternative course,* dan perluasan didalam *usage scenario*. Semua bagian tersebut merupakan faktor-faktor yang diperhatikan dalam mendesain antarmuka. Pada aplikasi yang menggunakan jaringan (LAN) aktivitas akan banyak dilakukan di komputer *client* sehingga dibutuhkan fitur aplikasi *client* yang kaya. Faktor-faktor yang berpengaruh adalah perangkat *client*, grafis, interaksi, *bandwith* jaringan, *disconnected client*, operasi CPU, penguncian basisdata, sumber daya lokal, keamanan yang terdiri dari otorisasi pemakai, autentikasi, komunikasi yang aman dan manajemen status serta aktivitas audit transaksi.

Dibawah ini digambarkan prototipe antarmuka aplikasi yang telah disesuaikan dengan kebutuhan yang ada pada *usecase* dan

usage scenario. Pada antarmuka dibawah ini digambarkan rancangan desain untuk *sales* yaitu halaman informasi produk.



Gambar 27. Desain Antarmuka Halaman Informasi Produk.

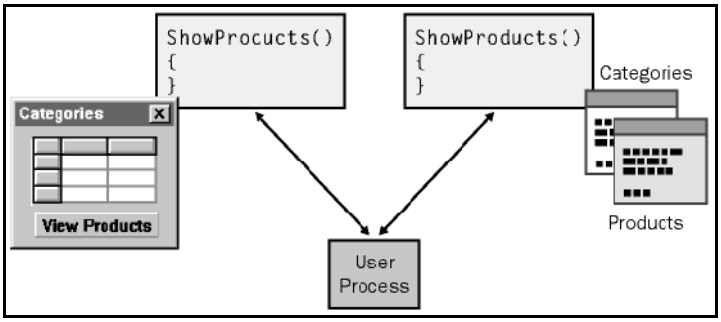
Sama dengan proses desain yang lain, hasil akhir dari aktivitas desain antarmuka meliputi :

- Tim proyek, pelanggan dan pemakai sepakat tentang petunjuk desain yang meliputi elemen yang digunakan didalam aplikasi.
- Desain haruslah mengandung deskripsi bagaimana antarmuka menyediakan umpan balik yang dibutuhkan seperti *progress bar* dan *user assistance*.
- Desain haruslah dapat diuji berdasarkan catatan histori dan di komparasi kembali dengan kondisi yang akan datang dari *usage scenario*.

Interaksi pemakai dengan antarmuka aplikasi adalah proses yang dapat diprediksi sebelumnya, misalnya interaksi pemakai ketika memasukkan detail produk, melihat total harga, memasukan detai pembayaran, memasukkan informasi alamat pengiriman. Proses-proses tersebut haruslah dapat dikelola dengan memperhatikan transisi antar proses yang terjadi dimana aktivitas satu proses menuju proses lainya. Untuk keperluan ini dibutuhkan komponen proses pemakai. Fungsi proses pemakai adalah :

- Status interaksi pemakai dalam waktu yang lama lebih mudah untuk di tetapkan. Memungkinkan pemakai untuk membatalkan dan memulai lagi proses yang telah dilakukanya. Misalnya pembeli menambahkan item belanja ke keranjang belanja online, oleh karena respon aplikasi sangat lama, maka pembeli menghubungi sales untuk menyelesaikan pesananya.
- Proses pemakai yang sama dapat digunakan lagi oleh berbagai antarmuka yang berbeda. Misalnya item yang dibeli oleh pelanggan, proses pembelianya dapat dilakukan online dengan aplikasi web dan proses tadi dapat digunakan lagi untuk aplikasi *offline* berbasis *window*.

Komponen proses pemakai diterapkan sebagai kelas yang memiliki metoda yang dapat dipanggil melalui antarmuka. Setiap metoda mengkapsulkan logika yang dibutuhkan untuk menjalankan aksi tertentu didalam proses pemakai.



Gambar 28. Antarmuka Pemakai dan Komponen Proses Pemakai.

Sebelum merancang komponen proses pemakai terlebih dahulu pisahkan komponen proses pemakai dari komponen antarmuka pemakai. Hal ini penting untuk menetapkan fungsi apa yang dapat dilakukan oleh antarmuka pemakai dan kebutuhan apa yang akan ditangani oleh komponen proses pemakai. Untuk memisahkanya ikuti langkah-langkah berikut ini :

1. Identifikasi proses bisnis atau proses-proses lainya dimana proses pemakai akan membantu dalam menyelesaikan tahapan proses bisnis. Untuk mengidentifikasi proses bisnis perhatikan diagram *sequence*, *usecase* dan *usage scenario*.
2. Identifikasi data yang diperlukan untuk proses bisnis.
3. Identifikasi status tambahan yang akan diperlukan untuk mengelola aktivitas pemakai dan membantu pengisian dan penangkapan data didalam antarmuka pemakai.
4. Membuat desain aliran secara visual dari proses pemakai dan bagaimana cara setiap element antarmuka pemakai menerima atau memberikan aliran kontrol.

Dalam merancang atau mendesain proses pemakai untuk aplikasi perhatikanlah petunjuk dibawah ini :

1) Menetapkan apakah pengelolaan komponen proses pemakai terpisah dari komponen antarmuka pemakai.
2) Tetapkan dimana proses pemakai akan disimpan.
3) Mendesain komponen proses pemakai secara berkesinambungan.
4) Gunakan penanganan kesalahan didalam komponen proses pemakai dan diteruskan ke komponen antarmuka pemakai.

BAGIAN 7

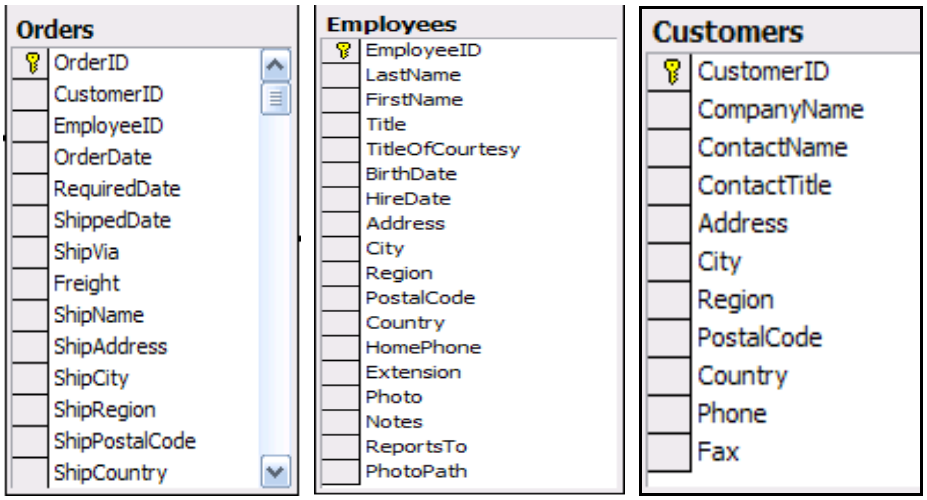
MENDESAIN LAPISAN DATA

7.1. Mendesain Penyimpanan Data

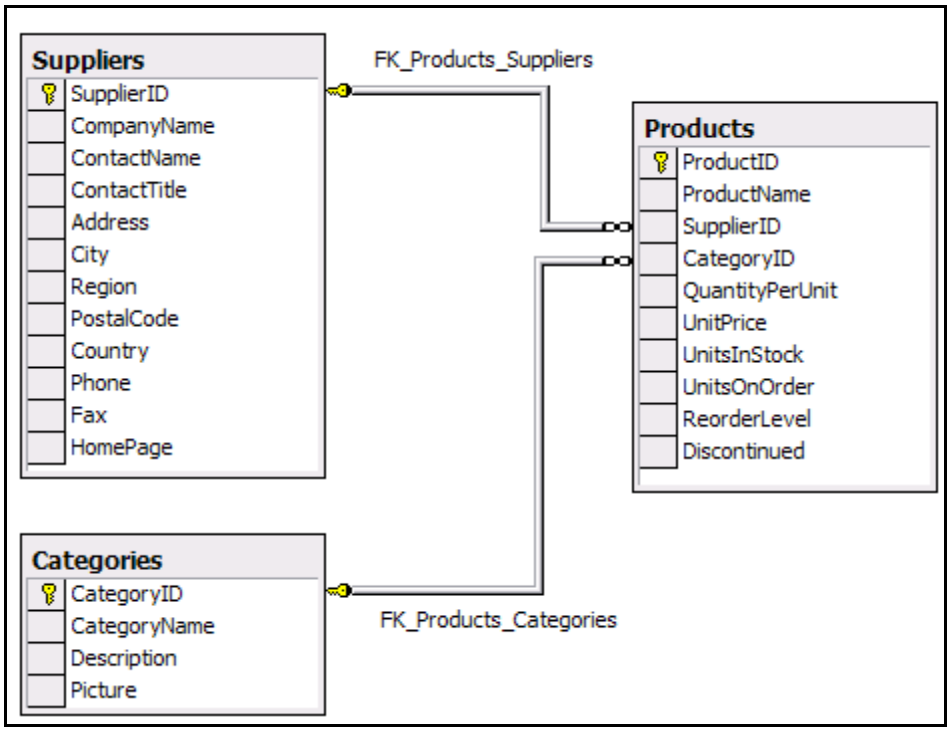
Selama fase *planning* dalam model proses MSF, tim proyek mendesain lapisan data dari solusi yang akan dirancang. Lapisan data dari sebuah solusi terdiri dari dua bagian yaitu penyimpanan data dan layanan data. Penyimpanan data umumnya membutuhkan sebuah basisdata dimana data diorganisasikan dan disimpan. Sebuah data dapat memiliki entitas dan atribut didalam suatu model data yang menjelaskan bagaimana data disusun dalam sebuah struktur data.

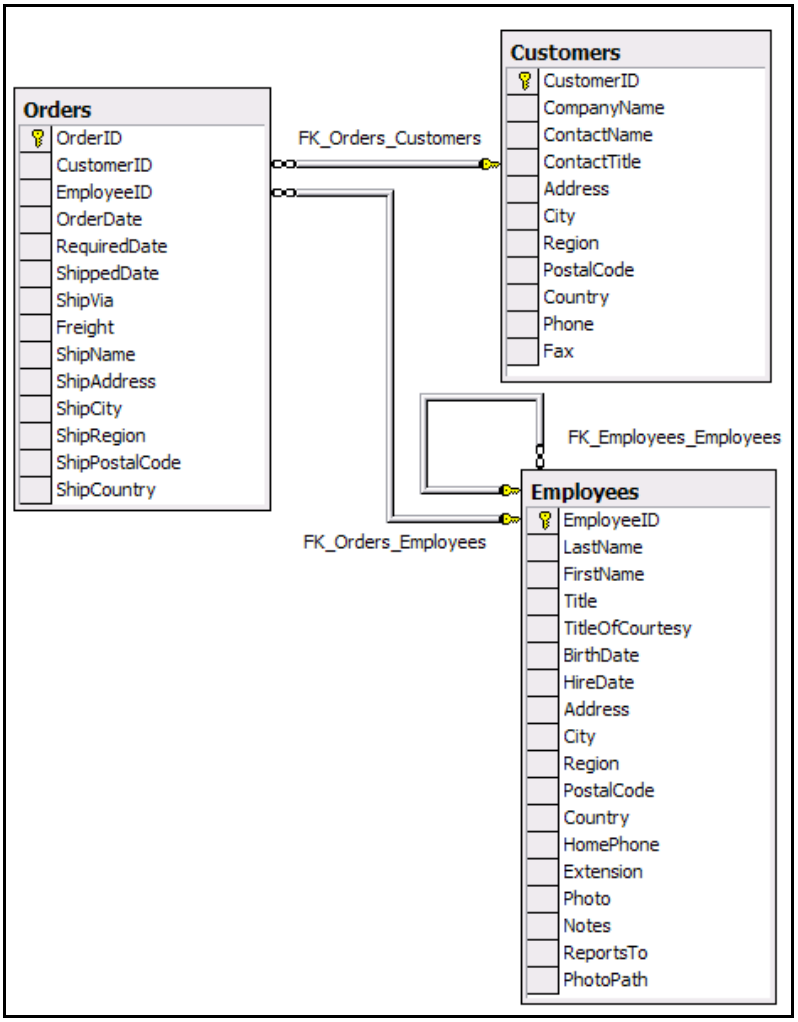
Basis data adalah definisi dari kumpulan nilai data yang diorganisasikan dengan cara tertentu. Skema basis data menentukan bagaimana data diorganisasikan didalam basis data tersebut. Ketika desain logikal tim mendiskripsikan entitas dan atribut yang akan disimpan didalam basis data, bagaimana pemakai mengakses data, memanipulasi dan melihat data. Sementara pada desain fisikal tim proyek merancang skema dari basis data yang menyediakan spesifikasi untuk membuat, membaca, memperbaharui, dan menghapus data didalam solusi. Skema basis data mendifinisikan entitas utama yang berhubungan dengan permaslahan bisnis, atribut dari entitas, dan relasi antar entitas. Umumnya objek-objek database dimodelkan didalam diagram relasi entitas (*entity relationship diagram*). Setiap *ERD* mengandung entitas, atribut dan relasi.

Contoh entitas dengan atributnya masing-masing



Contoh relasi entitas





Gambar 29. Diagram Relasi Entitas.

7.1.1. Entitas dan Atribut.

Entitas merupakan objek yang menyimpan informasi, misalnya produk, order dan pelanggan. Mengidentifikasi entitas merupakan langkah awal dari perancangan basisdata. Pada sebuah *usecase* dapat ditemukan objek yang akan menjadi entitas, misalnya "karyawan membuat sebuah kontrak dengan pelanggan", maka diperoleh objek karyawan, kontrak dan pelanggan. Karyawan melakukan sebuah aksi di dalam sistem, pelanggan merupakan

penerima aksi yang dilakukan oleh karyawan, sedangkan kontrak merupakan kesepakatan antara karyawan dengan pelanggan.

Setelah mengidentifikasi entitas, selanjutnya tim mendefinisikan atribut. Atribut memiliki karakteristik :

1) Atribut mendiskripsikan entitas dari solusi. Misalnya atribut dari karyawan misalnya no_induk, nama, alamat, dan departemen.
2) Atribut ada hanya ketika di cantumkan didalam entitas. Misalnya atribut warna tidak menjelaskan sesuatu yang berarti apabila warna belum diterapkan didalam objek.
3) Atribut mendefinisikan kolom tabel didalam basis data yang memiliki tipe data, lebar data dan hubungan dengan atribut didalam tabel lain.

7.1.2. Tabel dan Kolom (*field*).

Tabel adalah representasi dari entitas didalam database relasional. Tabel menyimpan data yang beragam sesuai kebutuhan bisnis. Data didalam tabel disimpan didalam baris (*row*) atau disebut *record*. Setiap *record* didalam tabel harusalah *unique* (hanya ada satu). *Record* didalam database dapat dimanipulasi menggunakan XML (*extensible markup language*), XML juga dapat digunakan untuk mentransmisikan data diantara basis data, diantara perusahaan yang berbeda. Metoda tradisional dalam memanipulasi data relasional ialah mengunakan bahasa standar relasional basisdata yang bernama ANSI (*american national standard institute*) dimana mengacu pada SQL (*structure query language*). SQL merupakan operasi yang dijalankan didalam database dengan statemen yang mudah dibaca seperti statement *insert, update* dan *delete*.

Kolom (*column*) atau *field* merupakan tempat penyimpanan nilai data pada *record*. Pada setiap *field* atau *column* menyimpan sebuah nilai yang berbeda. Tipe data menjelaskan macam data didalam *field.* Tipe data digunakan untuk memverifikasi nilai yang dimasukkan didalam *field* merupakan data yang valid, selain itu jangkauan data juga menentukan validitas data. Tipe data dapat dibedakan menjadi dua macam yaitu tipe data yang didefinisikan oleh sistem, misalnya *integer, string, currency*, *binary.* Tipe data yang didefinisikan oleh pemakai misalnya di dalam SQL server pemakai dapat mendifinisikan tipe data *state* dengan lebar data dua digit.

7.1.3. Kunci (*key*).

Kunci merupakan bagian yang penting didalam database relasional. Kunci secara unik menidentifikasi setiap instan dari entitas didalam model data. Secara umum kunci dapat dibedakan menjadi dua macam yaitu kunci primer (*primary key*) dan kunci tamu (*foreign key*).

1) Kunci primer (*primary key*), mengidentifikasi atribut yang unik untuk setiap instan dari entitas. Misalnya SalesOrderId adalah *field* yang unik untuk setiap order.
2) Kunci tamu (*foreign key*), kunci yang menghubungkan dua tabel. Misalnya ProductID adalah kunci tamu didalam tabel SalesOrderDetail, namun ProductID adalah kunci primer didalam tabel MasterProduct dimana tabel SalesOrderDetail dapat mengacu tabel MasterProduct.

7.1.4. Relasi.

Relasi antar entitas dibuat dengan menambahkan kunci dari sebuah entitas tabel ke entitas tabel lainya dimana entitas diikat bersama oleh nilai kunci. Kardinalitas relasi dibedakan menjadi tiga macam :

1. Relasi satu ke satu (*one to one*).

Relasi yang menggambarkan instan dari sebuah entitas secara langsung dihubungkan ke instan entitas lain. Sebagai contoh setiap departement hanya dapat memiliki anggota departemen sebagai kepala departemen. Relasi ini dapat direpresentasikan :

- Sebagai satu tabel, dua entitas dapat menjadi satu tabel dan menggunakan kunci primer sebagai kunci komposit. Keuntungan dari model ini yaitu tidak diperlukan pengelolaan tabel secara terpisah sehingga dapat menghilangkan *statemen query* dengan perintah *join*, kapasitas penyimpanan menjadi lebih efisien. Kekurangan dari model ini adalah jika ada perubahan terhadap relasi sehingga keputusan untuk mengubah membutuhkan biaya.
- Sebagai dua tabel, dengan tetap memisahkan menjadi dua buah tabel, pada dengan menambahkan kunci primer dalam tabel pertama dan kunci tamu pada tabel kedua. Pada model ini berlaku relasi *parent-child* yaitu *parent* tabel memiliki kunci primer dan *child* tabel memiliki kunci tamu, sehingga *child* tabel memerlukan *parent* tabel untuk keberadaanya. Pada model ini hanya dibolehkan nilai yang unik untuk masing-masing kunci sehingga dapat memastikan bahwa satu

instan sebuah entitas tabel hanya berelasi dengan subuah intans tabel lain.

- Sebagai banyak tabel, keberadaan *parent* tabel tidak memerlukan relasi dengan instan tabel *child*. Sehingga tabel terpisah dan masing-masing memiliki kunci tamu untuk menerapkan relasi. Sebagai contoh sebuah perusahaan memberikan fasilitas mobil untuk karyawan, kemudian perusahaan memberikan jaminan asuransi kepada mobil dan karyawanya, jika tidak ada fasilitas mobil yang diberikan karyawan atau mobil perusahaan tidak digunakan oleh karyawan maka kebijakan mengenai asuransi mungkin menjadi tidak ada. Sehingga pada situasi ini entitas tabel yang terlibat adalah karyawan, mobil dan asuransi. Tabel karyawan dan tabel mobil masing-masing memiliki kunci primer, sementara tabel asuransi sebagai kunci tamu untuk relasi ke tabel mobil dan tabel karyawan.

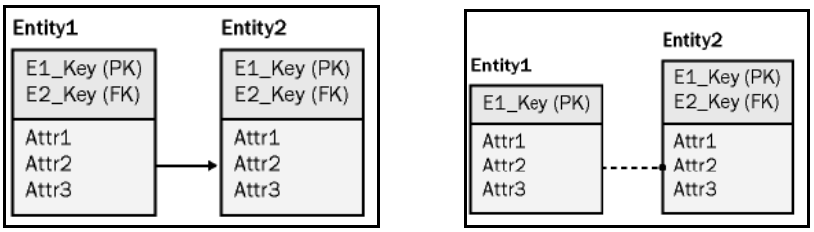
2. Relasi satu ke banyak (*one to many*).

Desain fisikan relasi satu ke banyak membutuhkan definisi tabel untuk entitas *parent* misalnya entitas pelanggan dan entitas *child* misalnya entitas order dimana keberadaan beberapa entitas *child* untuk setiap entitas *parent*. Misalnya relasi yang terjadi antara tabel produk dan tabel order, sebuah order dapat memiliki beberapa produk yang berelasi. Kunci tamu pada entitas produk menentukan eksistensi relasi.

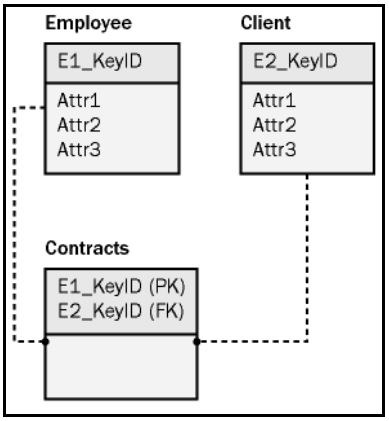
3. Relasi banyak to banyak (*many to many*).

Relasi yang terjadi antara beberapa tabel sehingga menunjukkan hubungan relasi dari satu entitas dengan entitas lain

dalam jumlah banyak. Sebagai contoh seorang karyawan dapat memiliki kontrak dengan beberapa pelanggan dan seorang pelanggan dapat memiliki kontrak dengan beberapa karyawan perusahaan. Sehingga setiap kunci primer digunakan sebagai kunci tamu didalam entitas tabel kontrak secara terpisah.



Gambar 30. Relasi Satu ke Satu dan Satu ke Banyak.



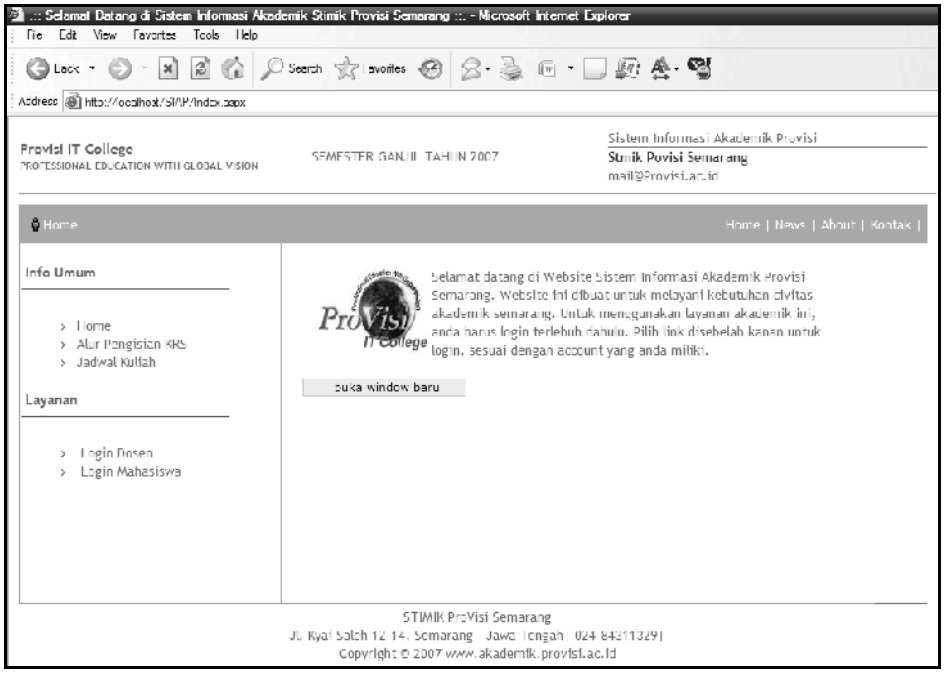
Gambar 31. Relasi Banyak ke Banyak.

7.1.5. Mengoptimalkan Akses Data.

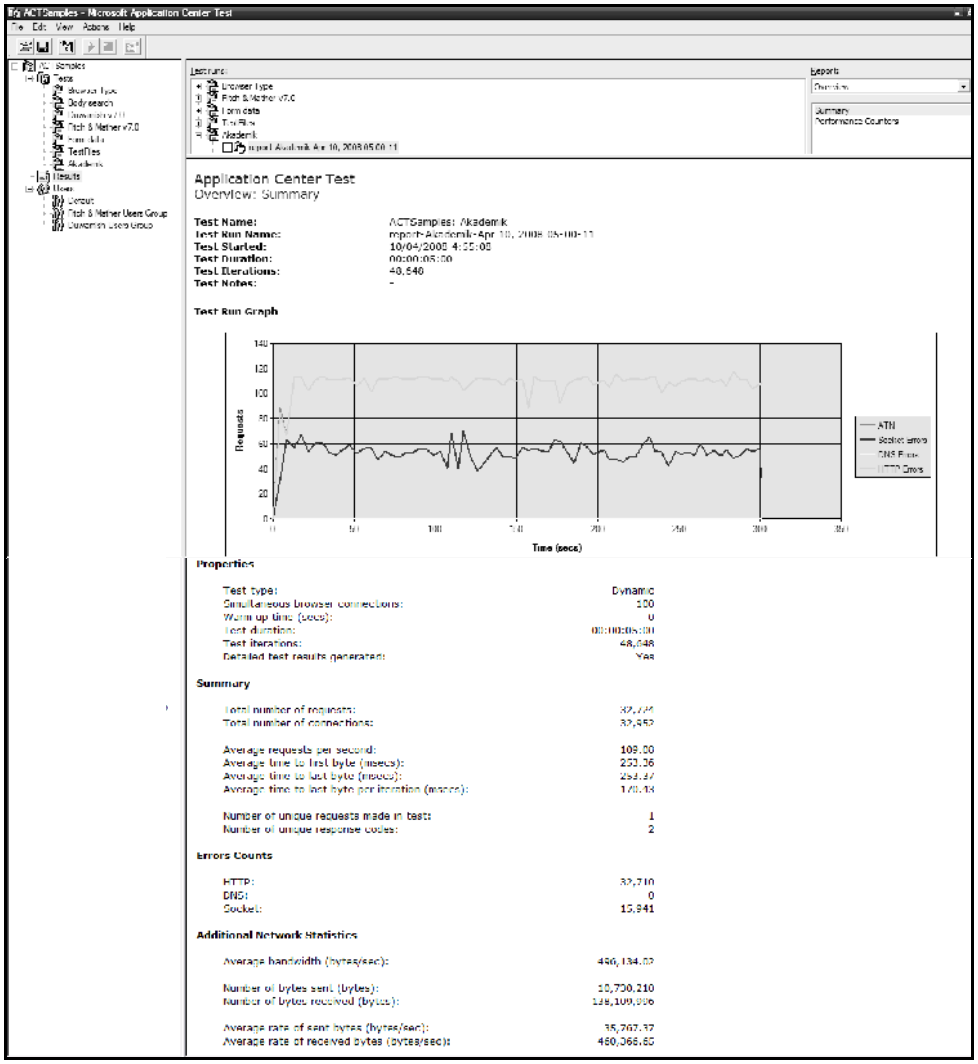
Optimalisasi performa aplikasi dan database sangat dibutuhkan untuk meningkatkan unjuk kerja solusi secara keseluruhan. Dalam mengoptimalkan aplikasi perlu mempertimbangkan petunjuk-petunjuk dalam penulisan kode program :

1) Meminimalkan *roundtrip* permintaan terhadap hasil yang ditampilkan.
2) Meminimalkan sejumlah data yang terkandung didalam hasil.
3) Mengurangi pengaksesan bersama untuk merubah *record* dan penanganan terhadap konflik secara efisien.
4) Berhati-hati mengevaluasi *tradeoff* antara pengelolaan data yang dihasilkan di *client* atau di *server*, khususnya aplikasi berbasis web.

Aplikasi juga perlu di uji tekanan, dapat menggunakan perangkat pengujian seperti Microsoft Aplication Stress Tool dan kemudian mengeksekusi aplikasi. Sebagai contoh aplikasi web akademik berikut ini.



Gambar 32. Apliksi Akademik Berbasis Web



Gambar 32. Hasil Pengujian Menggunakan ACT.

Aplikasi yang menggunakan database tentunya muncul aktivitas-aktivitas transaksi sehingga perlu penggunaan transaksi secara bijaksana yaitu konsumsi waktu transaksi yang pendek dan terjadi jika pada saat dibutuhkan saja terutama apabila aplikasi didistribusikan pada area-area yang berbeda. Pada aplikasi *client-server* dibutuhkan komunikasi data secara efektif didalam batasan

aplikasi dan batasan proses yang sangat berpengaruh pada performa aplikasi.

Basis data merupakan lapisan aplikasi yang paling bawah. Optimalisasi terhadap basis data sangat dibutuhkan agar aplikasi efektif ketika mengakses data. Teknik-teknik untuk mengoptimalkan basis data adalah :

a. Mengindeks data.

Ketika aplikasi mengakses dan memperbaharui basis data dibutuhkan pengurutan data sehingga *DBMS* dapat dengan cepat melakukan pengaksesan data. Indeks data disusun seperti *tree* dan digunakan untuk mengelola data yang telah diurutkan. Perintah *query* yang dijalnkan pada data berindeks lebih cepat dan lebih efisien dibanding *query* pada data yang tidak berindeks. Sehingga keuntungan dari pengaksesan data berindeks adalah akses data cepat dan tercapainya integritas data yaitu data yang tersimpan unik didalam *record*. Indeks dalam basis data dapat dibedakan menjadi *clustered dan nonclustered* indeks.

- *Clustered* indeks secara fisik akan melakukan pengurutan ulang data didalam tabel untuk menyesuaikan urutan indeks. *Clustered* indeks memiliki performa yang tinggi untuk operasi pembacaan data. Indeks jenis ini hanya dibatasi satu saja dalam DBMS, sebagai contoh didalam tabel SalesOderDetail menggunakan *field* SalesOrderId dan LineNumber sebagai *clustered* indeks.
- *Nonclustered* indeks secara sederhana mengelola informasi indeks dari tabel kecil dengan sebuah *column* dan kelompok *column*. Indeks jenis ini diperbolehkan lebih dari satu indeks.

b. Pemartisian data.

Pemartisian data dapat meningkatkan kecepatan akses data didalam basisdata. Pemartisian data dibedakan menjadi dua macam yaitu pemrtisian vertikal dan pemartisian hosrisontal.

- Partisi horisontal, dengan membagi tabel yang memiliki sejumlah record menjadi dua buah tabel dengan record yang berbeda tetapi memiliki *column* yang sama sehingga setiap tabel mengandung subset data. Sebagai contoh sebuah tabel mungkin mengandung semua data pelanggan dengan nama diawali dengan huruf "A" sampai "M" dan tabel lain mengandung data yang diawali dengan huruf "N" sampai "Z".
- Partisi vertikal, dengan membagi tabel yang mengandung *column* kedalam beberapa tabel yang mengandung *record* atau baris dengan pengenal unik yang sama. Sebagai contoh sebuah tabel mengandung data berstatus *read-only* sementara data lainya dapat di perbaharui (*not read-only*), pada data lingkungan data yang besar partisi vertikal dilakukan dengan memperluas database pada sejumlah server untuk distribusi data.

7.1.6. Normalisasi Data.

Normalisasi adalah proses yang secara progresif menyaring model logikal untuk mengeliminasi data yang sama didalam basis data. Aktivitas normalisasi data dilakukan dengan membagi data menjadi dua atau lebih tabel serta menentukan relasi antar tabel. Normalisasi dapat dilakukan sampai lima level bentuk normal,

sehingga diperoleh tabel dengan tidak ditemukan lagi duplikasi data. Aktivitas normalisasi basis data dilakukan adalah :

1. Meminimalkan duplikasi informasi.

   Basis data yang ternomalisasi mengandung sedikit duplikasi data dibanding basis data yang tidak dinormalisasi.

2. Mengurangi inkonsistensi data.

   Normalisasi dapat mengurangi konsistensi dengan mengelola relasi antar tabel. Sebagai contoh nomor telepon pelanggan tersimpan dalam beberapa tabel atau didalam beberapa *record* didalam tabel yang sama, ketika nomor telepon dirubah nomor telepon tidak akan berubah pada tabel lain. Berbeda jika nomor telepon disimpan pada satu tabel, maka integritas data terjaga.

3. Kecepatan dalam memodifikasi data (*insert*, *update* dan *delete*).

   Normalisasi mempercepat proses modifikasi data didalam basis data. Sebagai contoh penghapusan informasi nama pelanggan dan alamat pelanggan pada tabel *invoice*, data yang dihapus tidaklah hilang karena data berada di dalam tabel *client.*

Aktivitas normalisasi data akan diuraikan secara bertahap secara berurutan, sehingga akan menghasilkan data normal.

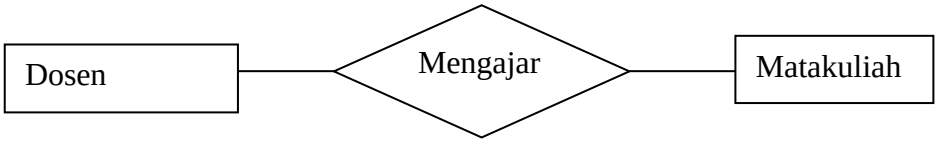
Didalam aktivitas normalisasi data didalam tabel diawali dengan mendefinisikan entitas-entitas dan atribut-atribut sebagai penyusun tabel didalam basisdata. Dibawah ini merupakan contoh entitas-entitas pada sebuah perguruan tinggi dengan contoh atribut.

Entitas		**Atribut**
Mahasiswa	1	***NIM (Primary Key)***
	2	Nama
	3	Tempat/Tanggal Lahir

	4	Agama
	5	Jenis kelamin
	6	Alamat
Dosen	1	***NIP (Primary Key)***
	2	Nama
	3	Jurusan
	4	Bidang Ilmu
	5	Alamat
Matakuliah	1	***Kode Matakuliah (Primary Key)***
	2	Nama Matakuliah
	3	Jumlah SKS
	4	Semester
	5	Prasarat

Setelah entitas-entitas dan attribut beserta kuncinya disusun, langkah selanjutnya adalah menentukan relasi antar entitas. Seperti yang telah dibahas sebalumnya bahwa relasi antar entitas dapat dibedakan menjadi tiga macam yaitu relasi satu ke satu, relasi satu ke banyak atau sebaliknya dan relasi banyak ke banyak. Dalam menentukan relasi maka dipertimbangkan hubungan antar entitas pada keadaan nyata, misalnya antara dosen dengan matakuliah, mahasiswa dengan matakuliah atau mahasiswa dengan dosen. Jika antara satu entitas berhubungan dengan entitas lainya maka hubungan tersebut dapat dinyatakan entitas baru sehingga ditentukan pula atribut dan atribut kuncinya. Misalnya relasi antara entitas dosen dan matakuliah akan menghasilkan sebuah entitas baru yaitu jadual. Dalam kondisi nyata bahwa seorang dosen pada setiap semester akan mengampu matakuliah tertentu, sehingga jika dua antitas ini berelasi maka akan dahasilkan entitas baru yaitu entitas

yang menyimpan informasi waktu dan tempat perihal kapan dan dimana matakuliah tersebut diajarkan oleh dosen yang bersangkutan. Dalam model *entity relationship diagram* relasi tersebut digambarkan sebagai berikut :



Entitas baru yang terbentuk ditentukan atribut-atribut bagi entitas Jadual.

Entitas		**Atribut**
Jadual	1	***NIP (Foreign Key)***
	2	***Kode Matakuliah (Foreign Key)***
	3	Hari
	4	Pukul
	5	Ruang
	6	Kelompok

Aturan-aturan dalam normalisasi dapat diartikan dalam istilah bentuk normal. Sebuah entitas dapat dikatan dalam bentuk normal apabila telah memenuhi aturan-aturan dalam normalisasi. Proses pembentukan data normal dilakukan secara bertingkat dari level normal pertama (1nf), kedua (2nf), ketiga (3nf), keempat (4nf) dan bentuk normal kelima (5nf). Pada normalisasi level ketiga sesungguhnya rancangan basisdata dapat dikatakan sudah baik.

1. Normalisasi Pertama (1NF).

Tahapan-tahapan dalam normalisasi di awali dengan normalisasi pertama. Tahapan normal pertama tabel seharusnya memiliki kriteria :

a. Tabel harus terbentuk dua dimensi dan diorganisasikan didalam baris dan kolom.
b. Setiap baris dan kolom harus mengandung sebuah nilai.
c. Setiap kolom harus memiliki arti (bernilai) tunggal.

Entitas Mahasiswa dan Atributnya Sebelum Normalisasi

Entitas		**Atribut**
Mahasiswa	1	***NIM (Primary Key)***
	2	Nama
	3	Tempat/Tanggal Lahir
	4	Agama
	5	Jenis kelamin
	6	Alamat
	7	Mata Kuliah
	8	SKS
	9	Nilai
	10	Waktu Kuliah
	11	Dosen Pengampu

Entitas Mahasiswa Menyimpan Data Record Sebelum Normalisasi

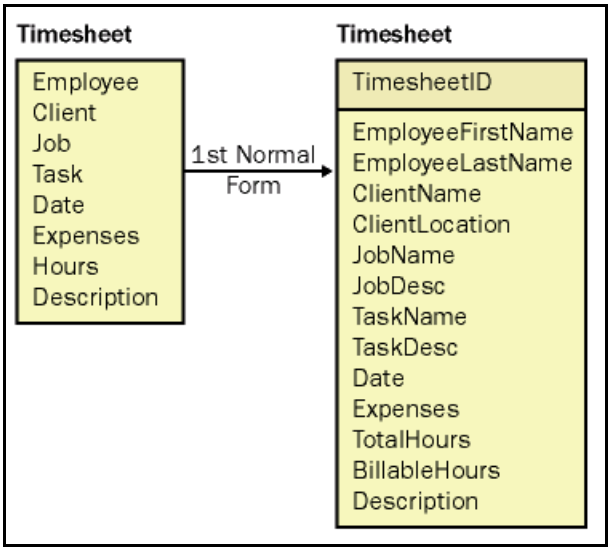
Nim	**Nama**	**Tempat Tanggal Lahir**	**Agama**	**Jenis Kelamin**	**Alamat**	**Mata Kuliah**	**SKS**	**Nilai**	**Waktu Kuliah**	**Dosen Pengampu**
001	Alfred Tio	Semarang 12/12/76	Islam	Laki-laki	Candi Lama 20	Rekayasa Perangkat Lunak	2	80	Senin 11.00-13.00 Ruang Teori 1	Migunani, S.Kom
002	Bajau Roni	Jakarta 11/11/78	Kristen	Laki-laki	Cempedak Utara 34A	Rekayasa Perangkat Lunak	2	85	Senin 11.00-13.00 Ruang Teori 1	Migunani, S.Kom
003	Bajau Roni	Jakarta 11/11/78	Kristen	Laki-laki	Cempedak Utara 34A	Pemrograman Web	4	78	Selasa 08.00-09.30 Ruang	Migunani, S.Kom

									Teori 1	
004	Muna Farida	Bandung 01/04/77	Islam	Wanita	Jend. A. Yani 20	Basisdata	4	90	Rabu 11.00-13.00 Ruang Teori 2	Samsudin, S.Kom

Pada tabel mahasiswa diatas atribut Nama, Tempat/Tanggal Lahir dan Waktu Kuliah masih bias dan belum memiliki arti tunggal. Sehingga atribut-atribut tersebut harus memiliki arti (bernilai) tunggal menjadi bentuk normal pertama (1nf).

Entitas		**Atribut**
Mahasiswa	1	***NIM (Primary Key)***
	2	Nama Awal
	3	Nama Akhir
	4	Tempat Lahir
	5	Tanggal Lahir
	6	Agama
	7	Jenis kelamin
	8	Alamat
	9	Mata Kuliah
	10	SKS
	11	Nilai
	12	Hari Kuliah
	13	Jam Kuliah
	14	Ruang Kuliah
	15	Dosen Pengampu

Perhatikanlah contoh entitas timesheet berikut ini, sesudah normalisasi yang pertama terlihat bahwa nilai atribut memiliki arti (nilai) tunggal.



Gambar 33. Bentuk Normalisasi Pertama (1NF).

Pada tabel Timesheet diatas (gambar sebelah kiri) belum ternomalisasi NF1 karena belum memiliki pengenal yang unik dan nilai pada kolom masih dapat memiliki berbagai arti. Pada normalisasi yang pertama atribut *Employee* dapat dibagi menjadi dua atribut yang berbeda yaitu *Firstname* dan *Lastname*. Demikian juga untuk atribut lainya masih dapat dibagi menjadi atribut yang berbeda.

2. Normalisasi Kedua (2NF).

Normalisasi tahap ke dua memiliki kriteria sebagai berikut :

1. Menghilangkan data redundan (nilai berulang) didalam entitas tabel.
2. Pindahkan atribut yang bergantung pada *primarykey* didalam tabel yang berbeda untuk *primarykey* yang berbeda.
3. Menggabungkan informasi dalam atribut jika dimungkinkan.

Entitas		Atribut
Mahasiswa	1	***NIM (Primary Key)***
	2	Nama Awal
	3	Nama Akhir
	4	Tempat Lahir
	5	Tanggal Lahir
	6	Agama
	7	Jenis kelamin
	8	Alamat
	9	Jurusan

Entitas		Atribut
Matakuliah	1	***Kode Matakuliah (Primary Key Baru)***
	2	Nama Matakuliah
	3	SKS
	4	Semester
	5	Jurusan
	6	Prasarat

Entitas	1	***NIP (Primary Key)***
Dosen	2	Nama
	3	Jurusan
	4	Bidang Ilmu
	5	Alamat
	6	Jabatan Fungsional

Entitas	1	***Kode Ruang (Primary Key)***
Ruang Kuliah	2	Nama Ruang
	3	Jenis Ruang
	4	Kapasitas

Entitas	1	***NIM (Foreign Key)***
Nilai	2	***Kode Matakuliah (Foreign Key)***
	3	Nama Matakuliah
	4	SKS
	5	Nilai Angka
	6	Nilai Huruf

Entitas	1	***Kode Matakuliah (Foreign Key)***
Jadual	2	***NIP (Foreign Key)***
	3	***Kode Ruang(Foreign Key)***
	4	Nama Matakuliah
	5	Nama Dosen
	6	Nama Ruang
	7	Hari
	8	Pukul

3. Normalisasi Ketiga (3NF)

   Normalisasi tahap ke tiga memiliki kriteria sebagai berikut :

   a. Hilangkan kolom (atribut) yang tidak bergantung pada nilai *primarykey*.
   b. Pindahkan kolom (atribut) yang secara tidak langsung berhubungan dengan entitas ke tabel lain.
   c. Kurangi atau hilangkan anomali untuk pembaharuan dan penghapusan data.
   d. Verifikasi terhadap redundansi (data rangkap) yang tersisa.

Entitas Nilai memenuhi bentuk 2nf tetapi belum memenuhi 3nf

Entitas	1	***NIM (Foreign Key)***
Nilai	2	***Kode Matakuliah (Foreign Key)***
	3	Nama Matakuliah
	4	SKS
	5	Nilai Angka
	6	Nilai Huruf

Dalam normal 3nf menjadi

Entitas		**Atribut**
Mahasiswa	1	***NIM (Primary Key)***
	2	Nama Awal
	3	Nama Akhir
	4	Tempat Lahir
	5	Tanggal Lahir
	6	Agama
	7	Jenis kelamin
	8	Alamat
	9	Jurusan

Entitas		**Atribut**
Matakuliah	1	***Kode Matakuliah (Primary Key)***
	2	Nama Matakuliah
	3	SKS
	4	Semester
	5	Jurusan
	6	Prasarat

Entitas	1	***NIM (Foreign Key)***
Nilai	2	***Kode Matakuliah (Foreign Key)***
	5	Nilai Angka
	6	Nilai Huruf

Entitas Jadual memenuhi bentuk 2nf tetapi belum memenuhi 3nf

Entitas	1	***Kode Matakuliah (Foreign Key)***
Jadual	2	***NIP (Foreign Key)***
	3	***Kode Ruang(Foreign Key)***
	4	Nama Matakuliah
	5	Nama Dosen
	6	Nama Ruang
	7	Hari
	8	Pukul

Dalam normal 3nf menjadi

Entitas	1	***NIP (Primary Key)***
Dosen	2	Nama
	3	Jurusan
	4	Bidang Ilmu
	5	Alamat
	6	Jabatan Fungsional

Entitas	1	***Kode Ruang (Primary Key)***
Ruang Kuliah	2	Nama Ruang
	3	Jenis Ruang
	4	Kapasitas

Entitas		Atribut
Matakuliah	1	***Kode Matakuliah (Primary Key)***
	2	Nama Matakuliah
	3	SKS
	4	Semester
	5	Jurusan
	6	Prasarat

Entitas	1	***Kode Matakuliah (Foreign Key)***
Jadual	2	***NIP (Foreign Key)***
	3	***Kode Ruang(Foreign Key)***
	7	Hari
	8	Pukul

Pada normalisasi tahap tiga telah diperoleh tabel dengan tidak ditemukan redundensi data *field* lagi.

7.1.7. Integritas Data.

Integritas data mengacu pada konsistensi dan akurasi data. Integritas data didalam basisdata dapat dikategorikan dalam tiga macam yaitu integritas domain, integritas entitas dan integritas referensial.

a) Integritas Domain

Integritas domain menetapkan sekumpulan nilai data yang valid dan menetapkan apakah nilai kosong (*Null*) diijinkan. Integritas ini ditekankan pada pengecekan keabsahan dan penolkan terhadap tipe data tertentu, format tertentu dan jangkauan nilai yang mungkin didalam sebuah field. Misalnya penetepan besaran diskon antara 5%

sampai dengan 12% sehingga nilai diskon diluar jangkauan diatas tidak valid.

b) Integritas Entitas

Integritas entitas mengharuskan setiap baris (*record*) didalam tabel merupakan pengenal bersifat unik yang biasa disebut dengan nilai *primarykey*. Dengan menerapkan integritas entitas dan domain akan membantu memastikan bahwa setiap entitas pada desain fisikal dapat dikelola dengan baik.

c) Integritas Referensial

Integritas ini untuk memastikan bahwa dua buah tabel yang berelasi haruslah memiliki hubungan yang valid antara tabel satu dengan lainya. Dengan menetapkan kunci primer pada sebuah tabel *parent* dan menetapkan kunci asing pada tabel *child*.

BAGIAN 8

MENDESAIN KEAMANAN PERANGKAT LUNAK

8.1. Sistem Kemanan Secara Umum.

Sebuah sistem komputer dapat memiliki kelemahan (*vulnerability*) yang memungkinkan pemakai non otoritatif dapat menggunakan sistem komputer tanpa autentikasi atau bahkan melakukan hal-hal negatif yang bertujuan untuk menghilangkan layanan sistem. Dengan menerapkan mekanisme keamanan sistem seperti penggunaan *firewall*, *proxies*, *secure channel,* dan skema *authentication* akan meningkatkan sistem keamanan.

Pemakai non otoritatif (*malicious user*) dapat menggunakan metoda eksploitasi yang beragam untuk menghancurkan sistem. Kelemahan sistem merupakan lubang keamanan yang akan dijadikan titik awal untuk menghancurkan sistem secara menyeluruh dan

meluas. Beberapa kelemahan sistem atau kelemahan aplikasi yang dirancang meliputi :

1. Kata kunci yang lemah (*weak password*), dengan *password* yang lemah memungkinkan penyerang dapat mengakses tidak hanya satu komputer namu dapat meluas pada keseluruan jaringan dimana komputer-komputer terhubung.
2. Perangkat lunak yang salah konfigurasi (*misconfigured software*), sering kali karena kesalahan dalam konfigurasi perangkat lunak akan menyebabkan potensi kelemahan yang muncul. Jika akun pengguna lokal dikonfigurasi dengan memberikan hak akses yang lebih maka pemakai non otoritatif dapat memanfatkan sistem dan menjalankan aksi yang berbahaya didalam sistem.
3. Rekayasa sosial (*social engineering*), jika pemakai tidak menyadari bahwa sebuah kata kunci *(password*) yang dimilikinya merupakan hal yang sangat rahasia, dengan rekayasa sosial sebuah kata kunci dapat diungkapkan kepada pihak lain yang memiliki maksud negatif.
4. Koneksi internet (*internet connection*), instalasi standar web server IIS versi 5.0 membuka layanan dan *port* yang diperlukan untuk pengoperasian aplikasi tertentu. Misalnya koneksi modem melewatkan *firewall* yang melindungi jaringan dari penyusup luar.
5. Transfer data yang tidak terenkripsi, apabila data dikirimkan antara komputer server dan komputer pemakai dalam format teks standart, akan sangat mungkin data teks ditangkap, dirubah dan ditransmisikan melalui jaringan.

6. Memori penyangga yang berlebih, pemakai dapat melakukan pemenuhan buffer sehingga aplikasi atau sistem operasi menjadi *crash*.
7. Injeksi perintah query (*SQl Injection*), dengan memodifikasi statemen SQL dan menjalankan aksi yang negatif.
8. Kerahasisan didalam kode (*secret in code*), beberapa masalah kemanan komputer yang di rancang pemakai non otoritatif dapat menemukan rahasia yang ditanamkan didalam kode seperti password dan kunci enkripsi.

Berdasarkan kelemahan-kelemahan sistem yang memungkinkan terjadinya kerusakan sistem maka dibutuhkan strategi dalam merancang keamanan sistem. Strategi perancangan kemanan sistem hendaknya memenuhi prinsip-prinsip berikut ini :

1. Mempercayakan konfigurasi keamanan yang telah teruji dan menjamin sistem keamanan dibandingkan merancang sendiri sistem keamanan.
2. Jangan mengijinkan dan mempercayai masukan eksternal apapun, dengan memvalidasi data yang dimasukkan oleh pemakai atau oleh *service* tertentu.
3. Mengasumsikan bahwa sistem eksternal tidaklah aman. Jika aplikasi menerima data sensitive yang tidak terenkripsi dan berasal dari sistem eksternal, asumsikan bahwa informasi telah disepakati.
4. Menerapkan kewenangan akses yang sangat minimal. Janganlah mengaktifkan atribut pada akun layanan yang berlebihan untuk pemakai biasa.

5. Mengurangi komponen dan data yang tersedia, resiko akan meningkat dengan adanya sejumlah komponen dan data yang disediakan oleh aplikasi, agar minimal maka disediakan fungsi-fungsi yang hanya dibutuhkan saja.
6. Menggunakan mode keamanan standard, dengan tidak mengaktifkan layanan-layanan, hak pengguna dan teknlogi lain yang tidak secara eksplisit dibutuhkan.
7. Menerapkan enkripsi data dengan algorithma yang sulit dipecahkan.

8.2. Merencanakan Keamanan Aplikasi.

Sebelum merencanakan sistem keamanan pada aplikasi haruslah dipahami bermacam-macam ancaman yang mungkin terjadi pada aplikasi. Merencanakan sistem keamanan aplikasi meliputi :

1. Mengidentifikasi ancaman-ancaman aplikasi (*threat modelling*). Aktivitas ini merupakan aktivitas yang sangat penting dalam merencanakan kemanan aplikasi karena sebelum mengidentifikasi ancaman maka kebijakan keamanan terhadap aplikasi tidak dapat ditetapkan. Dengan mengumpulkan informasi mengenai aset apasaja dari sebuah organisasi yang akan di proteksi dan ancaman apa saja untuk setiap aset yang dimiliki.
2. Menetapkan kebijakan keamanan sebagai tindakan prefentif atau meminimalkan adanya ancaman. Setelah ancaman-ancaman dapat diidentifikasi maka ancaman-ancaman tersebut dikategorikan dan mendefinisikan strategi untuk setiap ancaman.

Keamanan aplikasi gagal karena kebijakan keamanan tidak direncanakan pada awal mula mendesain dan mengembangkan aplikasi. Merencanakan dan menerapkan fitur keamanan haruslah

selama pengembangan aplikasi. Didalam model MSF tahap-tahap kebijakan keamanan ditetapkan sebagai berikut :

MSF Phase	***Security Initiative***
Envisioning	Mengumpulkan kebutuhan keamanan. Tim berbincang dengan pelanggan dan *stakeholder* untuk memperoleh informasi mengenai data-data dan operasi-operasi yang bersifat sensitif, hak akses setiap pengguna, bagaimana keamanan di kelola sesuai kebutuhan didalam dokumen *requirements*.
Planning	Menetapkan model ancaman untuk mengantisipasi ancaman-ancaman yang mungkin. Didalam dokumen spesifikasi fungsional, usulkan fitur-fitur keamanan untuk meringankan resiko-resiko ancaman terhadap kemanan.
Developing	Menerapkan fitur keamanan yang teridentifikasi didalam dokumen spesifikasi fungsional.
Stabilizing	Menjalankan uji kemanan. Melakukan revisi terhadap model ancaman jika ditemukan informasi baru yang belum teridentifikasi selama pengembangan, pengujian dan umpan balik dari pengguna.
Deploying	Monitoring ancaman-ancaman yang mungkin pada aplikasi.

8.3. Model STRIDE.

Model STRIDE merupakan teknik yang digunakan untuk mengidentifikasi dan mengatagorikan ancaman-ancaman keamanan seperti : *Spoofing identitiy*, *Tampering*, *Repudiation*, I*nformation disclosure, Denial of service, Elevation of Previlage*.

1. Spoofing identity.

Pengguna yang tidak diotorisasi dapat menggunakan aplikasi karena berhasil memperoleh *password* pengguna yang terotorisasi.

2. Tampering.

   Pengguna memperoleh akses kekomputer dan merubah operasi, konfigurasi dan data. Tampering dapat dibedakan menjadi dua macam yaitu *malicious* dan *Accidental*. *Malicious* terjadi jika pengguna yang tidak diotorisasi berhasil mengkases sedangkan *accidental* dikarenakan pengguna secara tidak hati-hati menghapus file didalam jaringan atau merubah basisdata padahal tidak memiliki hak untuk merubah.

3. Repudiation.

   Administrator tidak berhasil membuktikan pengguna yang jahat atau membuktikan aksi tertentu. Misalnya pengguna berhasil menyerang sistem dan sistem tidak dapat mencatat aktivitas tersebut.

4. Information disclosure.

   User yang didak diotorisasi melihat data-data yang bersifat privat, misalnya nomor kartu kredit dan tanggal kadaluarsanya.

5. Denial of service (DOS).

   Serangan yang menyebabkan sistem berhanti atau serangan yang membatasi akses ke sistem. DOS dapat menyebabkan sistem operasi komputer dan aplikasi dapat berheti, CPU terbebani dengan banyak proses, sistem memori di gunakan berlebih sehingga sistem operasi dan aplikasi menjadi sangat lambat, bandwidth jaringan berkurang.

6. Elevation previlage.

Pengguna memperoleh akses yang lebih tinggi dari previlege yang diberikan administrator, sehingga pengguna yang tidak sah dapat melakukan serangan.

Setelah ancaman-ancaman teridentifikasi selanjutnya adalah bagaimana merespon adanya ancaman tersebut. Merespon adanya ancaman-ancaman keamanan aplikasi dapat dilakukan dengan :

1. Menginformasikan kepada pengguna tentang ancaman-ancaman yang mungkin. Ketika kelemahan sistem muncul ketika pengguna menjalankan aktivitas tertentu, maka dapat dinformasikan kepada pengguna bahwa aktivitas yang dilakukan dapat menimbulkan ancaman pada sistem.
2. Menghapus fitur. Jika pengguna mengetahui bahwa sebuah fitur akan mengantarkan pada resiko keamanan yang signifikan dan resiko tersebut tidak dapat dikurangi secara efektif. Sebaiknya mempertimbangkan aplikasi yang telah final.
3. Mengidentifikasi teknik meringankan (mitigasi) resiko. Jika pengguna memilih menambahkan fitur seharusnya ditetapkan juga teknik untuk meringankan resiko yang muncul yang dapat dikategorikan dalam beberapa teknik diantaranya adalah :
    a. Autentikasi dan otorisasi (*authentication and authorization*). Autentikasi adalah proses dimana sebuah entitas memferifikasi entitas lain siapa atau apa yang melakukan permintaan. Otorisasi adalah proses untuk menetapkan akses ke sumber daya.

b. Komunikai yang aman (*secure communication*). Harus dipastikan bahwa komunikasi diantara lapisan aplikasi aman ketika di data di transmisikan atau di muatkan didalam antrian. Untuk komunikasi yang aman dapat menggunakan teknologi antara lain *secure socket layer* (SSL) dimana data didalam chanel transmisi akan dienkripsikan ketika terjadi komunikasi antara klient dan server, IPSec digunakan untuk mengamankan data yang dikirim dari di buah komputer misalnya aplikasi server dan database server, Virtual Private Networking (VPN) digunakan untuk transport IP dari satu titik ke titik melalui jaringan intranet atau internet.
c. Kualitas layanan (*Quality of Service*). Kualitas layanan mengacu pada implementasi pembuatan profile didalam suatu pesan yang dikirimkan kedalam sistem.
d. Membatasi atau menghambat (*Throttling*). Membatasi jumlah pesan yang dikirimkan kedalam sistem.
e. Pemeriksaan (*Auditing*). Pemeriksaan adalah proses pengumpulan informasi tentang aktivitas pengguna, kejadian-kejadian penting yang akan disimpan dan dianalisis kemudian.
f. Penyaringan (*filtering*). Merupakan proses dalam mengintersepsikan dan menguji sebuah pesan yang dikirim didalam sistem.
g. Pembatasan hak akses. Menyediakan akses yang menimal untuk mengijinkan pemakai menyelesaikan tugasnya.

Dibawah ini adalah contoh teknik untuk mitigasi dengan menerapkan model STRIDE :

Mitigation technique	**Type of threat**
Authentication	S, D
Protect secrets	S, I
Audit trails	R
Do not store secrets	S, I
Privacy protocols	I
Authorization	T, I, D
Hashes	T
Message authentication codes	T
Digital signatures	T, R
Tamper-resistant protocols	T, R
Time stamps	R
Filtering	D
Throttling	D
Quality of service	D
Run with least privilege	E

BAGIAN 9

MELENGKAPI TAHAP PLANNING

9.1. Perencanaan Fitur Administratif.

Fitur administratif sangatlah penting bagi dukungan solusi dimana terbagi kedalam tiga macam yaitu *monitoring, data migration, dan licensing spicification*. *Monitoring* digunakan untuk memastikan

bahwa aplikasi berfungsi secara tepat dan menyediakan kondisi pada level yang optimal. Automated monitoring memungkinkan untuk identifikasi secara otomatis kondisi yang salah dan permasalahan sehingga membantu untuk mengurangi waktu yang dibutuhkan untuk mengembalikan kondisi dari kesalahan.

Rencana monitoring adalah mendifinisikan proses didalam lingkungan operasional yang akan memonitor solusi aplikasi. Rencana monitoring berisi apa saja yang akan dimonitor, dan bagaimana melakukan monitoring, dan bagaimana hasil dari monitoring dilaporkan dan dimanfaatkan. Elemen-elemen rencama monitoring terdiri dari :

1. Memonitor ambang batas sumber daya, dengan mengidentifikasi sumberdaya solusi yang akan dimonitor termasuk ambang batas nilai untuk setiap sumber daya. Misalnya ambang batas penggunaan sumberdaya prosesor adalah 80%.
2. Memonitor unjuk kerja, mendefinisikan ukuran unjuk kerja solusi dan komponen komponennya. Misalnya mencatat aktivitas login pemakai dan mencatat pula jumlah kegagalan login pemakai.
3. Menganalisa kecenderungan, mendefinisikan analisis dimana nilai data yang disimpan selama memonitor unjuk kerja. Misalnya memprediksikan sejumlah pengguna dalam menggunakan bagian-bagian dari solusi setiap harinya dan menentukan bagaimana solusi aplikasi akan dijalankan didalam sejumlah pemanggilan.
4. Pendeteksian kegagalan, menjelaskan bagaimana proses pengembangan, operasi, dan perawatan yang akan menggunakan spesifikasi fungsional dan kriteria penerimaan pemakai untuk

mendeteksi suatu kegagalan. Misalnya konsumen yang berbelanja online dapat melihat-lihat item dan memilihnya dari daftar item, jika daftar item tidak menampilkan produk maka kejadian ini akan di tandai.

5. Pendeteksian kesalahan, menjelaskan proses, metoda, sarana prasarana akan digunakan untuk mendeteksi dan mendiagnosa kesalahan solusi.

6. Pencatatan kejadian, menguraikan hasil pencatatan kejadian bagi sistem untuk *capturing* dan *reviewing* aplikasi secara signifikan dan kejadian-kejadian didalam sistem. Misalnya dengan menggunakan SQL Server untuk pencatatan kejadian-kejadian aplikasi.

7. Notifikasi, menguraikan bagaimana operasi yang dilakukan tim akan di informasikan ketika memonitor dan menelusuri eksepsi yang terdeteksi kesalahan solusi.

8. Tools, menguraikan sarana dan prasarana yang digunakan tim dapat digunakan untuk mendeteksi, mendiagnosa, membetulkan kesalahan dan untuk meningkatkan unjuk kerja aplikasi.

9.2. Perencanaan Fitur Migrasi Data.

Migrasi data adalah memindahkan atau merubah data dari format yang digunakan sebelumnya ke format data yang baru. Ketika solusi aplikasi di bangun sangatlah perlu untuk mengidentifikasi data yang digunakan pada aplikasi sebelumnya, sehingga aktivitas migrasi data dapat dilakukan dengan baik untuk aplikasi yang baru. Rencana-rencana migrasi data meliputi :

1. Strategi migrasi, menguraikan strategi yang akan menuntun proses migrasi.

2. Peralatan, mengidentifikasi peralatan yang digunakan untuk mendukung strategi migrasi.
3. Petunjuk migrasi, menjelaskan bagaimana melakukan tahap-tahap migrasi data.
4. Proses migrasi, menguraikan bagaimana proses migrasi dilakukan.
5. Pengujian lingkungan migrasi, menguraikan lingkungan pengujian diamana lingkungan pengujian ini merupakan replikasi lingkungan produksi.
6. Pembatalan konfigurasi ke keadaan sebelumnya ketika terjadi kesalahan.

9.3. Membuat Spesifikasi Lisensi.

Didalam pengembangan solusi aplikasi dibutuhkan perangkat lunak dan perangkat keras sebagai dukungan operasi. Dengan menetapkan spesifikasi perangkat lunak dan perangkat keras akan membantu dalam memastikan bahwa proses persetujuan pada setiap tahap perancangan dan memberikan waktu bagi vendor dalam mensuplai perangkat tersebut dengan tidak mengganggu penjadwalan. Bagian yang penting dalam menentukan spesifikasi pembelian perangkat lunak dan perangkat keras adalah masalah lisensi.

Diperlukan spesifikasi lisensi pada tahap pengembangan dan penyebaran solusi aplikasi. Selama pengembangan, tim akan menggunakan teknologi dan produk perangkat lunak tertentu yang telah dipilih sehingga haruslah dipastikan bahwa tim telah memiliki lisensi produk-produk yang digunakan.

9.4. Merencanakan Tahapan Kedepan (Pengembangan dan Penyebaran).

Sebelum tim memulai mengembangankan solusi sangatlah penting untuk memverifikasi lingkungan pengembangan dan lingkungan pengujian telah siap. Secara ideal merepresentasi lingkungan produksi, tetapi tim juga harus menyeimbangkan kebutuhan biaya pada setiap tahap pengembangan yang dilakukan.

Pada tahap perencanaan (*planning*) tim proyek membuat rencana master proyek dan penjadwalan proyek. Rencana pengembangan menguraikan proses pengembangan solusi. Dengan proses pengembangan yang terdokumentasi mengindikasikan bahwa tim telah melakukan diskusi dan telah menyetujui struktur dan arahan yang diterapkan selama proses pengembangan. Elemen-eleman proses pengememembangan adalah tujan pengembangan, strategi penyampaian keseluruhan solusi, pendekatan *tradeoff*, kunci utama desain, pengembangan dan lingkungan pengembangan, petunjuk dan standarisasi, kontrol kode dan pembuatan versi, proses pembangunan, komponen, perangkat menajemen pengembangan dan konfigurasi, *design patern*, pelatihan tim pengembang, dan dukungan tim pengembang.

9.5. Spesifikasi Teknis.

Spesifikasi teknis adalah sekumpulan dokumen referensi yang biasanya mengandung artifak dari desain fisikal yaitu spesifikasi class, model komponen, *metrics*, jaringan dan komponen teknologi. Selama tahap pengembangan (*developing*) tim menggunakan spesifikasi teknis untuk mendefinisikan dan jangkauan pekerjaan. Spesifikasi teknis mencakup definisi interface, *registry entries*, ukuran

byte yang dibutuhkan untuk installasi, *dynamic link library* (DLL), nama *assembly*. *Strong name*, kunci dan semua element yang berpengaruh pada tahap penyeberan dan distribusi. Dokumen spesfifikasi teknis mencakup elemen-elemen berikut ini :

1. *Architecture overview,* mendiskripsikan arsitektur yang akan diimplementasikan oleh solusi aplikasi.
2. *Object model,* mendeskripsikan model objek dari solusi dan fungsi-fungsinya.
3. *Interface*, mengandung kode dan detail metoda pada setiap antarmuka (*interface*) solusi.
4. *Code flow,* mendeskripsikan operasi setiap metoda didalam solusi.
5. *Error codes*, mendeskripsikan kode error yang digunakan untuk penaganan kesalahan pada solusi.
6. *Error logging*, mendeskripsikan bebagai kesalahan yang akan ditangani dan dicatat didalam solusi.
7. *Configuration,* menjelaskan bagaimana solusi aplikasi di register pada komputer tujuan.
8. *Supporting Documentation,* daftar dokumen yang menguraikan solusi yang mencakup spesifikasi fungsional dan lokasinya.
9. *Issues*, menjelaskan isu-isu yang diketahui perihal solusi yang umumnya mengenai tanggal untuk setiap spesifikasi pada sesi.

BAGIAN 10

TAHAP PENSTABILAN DAN PENYEBARAN SOLUSI

10.1. Tahap Penstabilan MSF.

Selama tahap pengujian telah dilakukan pada solusi aplikasi dengan fitur-fitur yang telah lengkap. Pengujian diawali dengan aktivitas seperti mendefinisikan kriteria sukses dan pendekatan pengujian selama fase envisioning dan membuat rencana pengujian pada tahap perencanaan. Ada dua tugas utama didalam tahap penstabilan yaitu :

1. Pengujian solusi, tim menerapkan rencana pengujian yang telah dibuat pada tahap perencanaan, dimana akan diuji pada fase pengembangan.
2. Melaksanakan pilot, tim memindahkan pilot untuk solusi aplikasi dari pengembangan untuk menguji solusi dengan pengguna sesungguhnya dan skenario yang nyata sebelum mendistribusi atau menyebarkan solusi aplikasi.

Dalam melakukan pengujian beberapa aspek yang perlu dipertimbangkan diantaranya yaitu :

1. Kriteria sukses, proyek harus terukur dengan kriteria-kriteria tertentu yang telah didefinisikan pada tahap *envisioning* dan *planning*.
2. *Zero-defect mind-set*, tim proyek konsisten untuk menghasilkan kualitas produk aplikasi.
3. Jenis pengujian, pengujian dapat dibedakan menjadi :
    a. *Coverage testing* : pengujian level rendah, misalnya ketika programer menuliskan kode maka akan diuji untuk memastikan bahwa solusi sesuai dengan spesifikasi fungsional. Dengan menerapkan prinsip sistem *buddy* dimana pengujian dilakukanoleh kolega (mempekerjakan teman sejawat) dan dapat berjalan dengan baik apabila memiliki skill yang sama untuk kebutuhan pengembangan.
    b. *Usage testing,* pengujian level tinggi yang seringkali dilaksanakan oleh pemakai potensial atau bagian dari kelompok. Pengujian ini sangat penting karena untuk memastikan bahwa isu dan bug yang berhubungan dengan unjuk kerja pemakai dapat ditangkap dan ditangani.
    c. *Testing term,* sebelum mempelajari lebih jauh perihal solusi sangatlah penting memahami terlebih dahulu terminologi pengujian.
    d. *Check-in test,* untuk memastikan kode yang diuji dapat diterima, dimana pengetesan ini dilakukan oleh developer dan tester sebelum kode di cek didalam sistem kontrol perubahan.

e. *Unit test,* pengujian bagian area internal (setiap modul) dimana *tester* memperoleh keuntungan berupa pengetesan secara otomatis didalam kode secara langsung.
f. *Functional test,* pengujian normal yang didefinisikan oleh tim pengembang dengan menerapkan pengujian *end-to-end.*
g. *Build verification test,* tujuan ini untuk mengidentifikasi kesalahan program selama proses *build* dengan pengujian *compilation-error dan runtime-erro.*
h. *Regression test,* proses pengujian yang berulang untuk kode yang baru ditulis atau diperbaharui dalam versi baru.
i. *Configuration test,* oleh karena solusi diinstal dan dikonfigurasi didalam komputer yang berbeda dan dengan cara yang berbeda.
j. *Compatibility test,* ada kebutuhan bagi solusi untuk di integrasikan dan dioperasikan dengan sistem yang sudah ada.
k. *Stress test,* dikenal dengan *load test* digunakan untuk mengidentifikasi isu atau *bug* yang merepresentasikan ketika solusi didalam pengembangan dengan tingkat tekanan yang tinggi.
l. *Performance test,* focus pada peningkatan unjuk kerja atau performasi test ini berhubungan dengan *stress test*.
m. *Documentation dan Help Test*, pengujian terhadap dukungan sistem berupa dokumentasi dan fitur bantuan (*help*) untuk mencari instruksi yang mungkin salah, teks dan gambar yang sudah kadaluarsa dan lainya.

n. *Alpha dan betha test*, pengujian kode alpha merujuk pada semua kode yang dihasilkan pada tahap pengembangan, sementara kode beta diuji pada tahap penstabilan.
o. *Parallel testing*, pengujian secara paralel dengan menguji solusi aplikasi yang ada dan aplikasi yang baru dalam rangka pengembangan .

10.2. Pengujian Pilot.

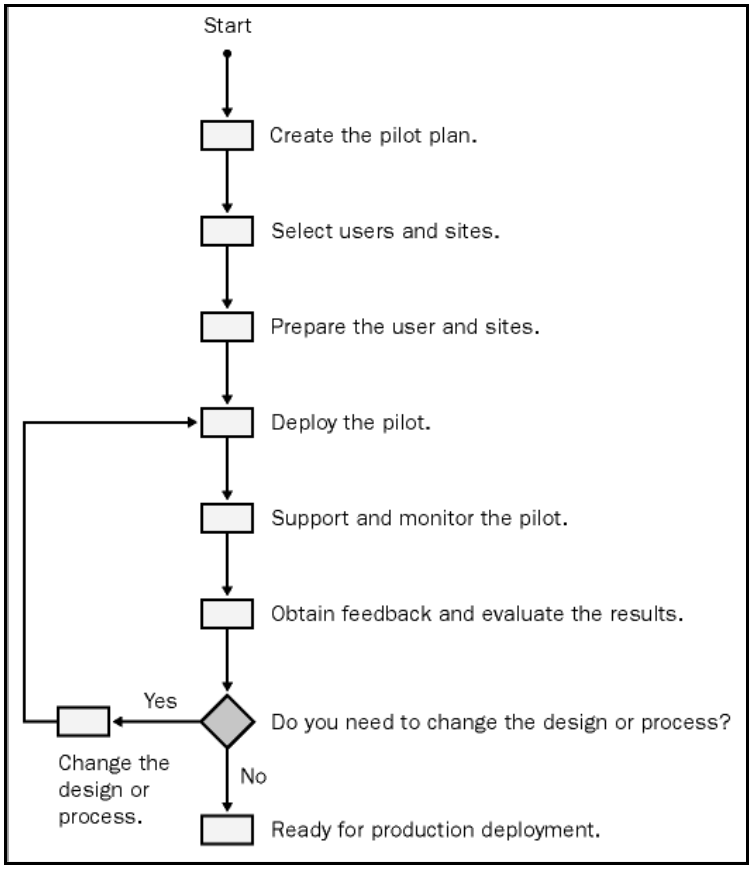
Pengujian pilot adalah pengujian solusi aplikasi pada lingkungan sesungguhnya. Dengan mencoba solusi dengan installasi, staff pendukung dan pengguna akhir. Tujuan utama dari pengujian pilot adalah mendemonstrasikan desain solusi pada lingkungan sebenarnya apakah telah sesuai dengan kebutuhan organisasi. Ada berbagai kondisi dengan jenis proyek yang berbeda seperti :

1. Pada perusahaan, pengujian pilot terdiri dari sekelompok pemakai atau beberapa server didalam data center server.
2. Didalam pengembangan web, pilot dapat menggunakan file yang dihosting pada sebuah server atau folder yang online di internet.
3. Vendor piranti lunak konvensional seperti Microsoft, biasanya produk yang sudah release kepada sekelompok pengguna yang telah mengadopsi produk.

Dalam mempersiapkan proyek pilot sebelum dijalankan memiliki tahapan-tahapan sebagai berikut :

1. Tim pengembang dan siapa saja yang berpartisipasi dalam pilot haruslah sepakat mengenai kriteria-kriteria sukses dari sebuah pilot.

2. Sebuah dukungan struktur dan isu-isu proses haruslah sudah siap. Proses ini biasanya membutuhkan staf pendukung yang telah di latih.
3. Untuk mengidentifikasi isu-isu dan mengkonfirmasikan proses pengembangan dapat berjalan.
4. Memerlukan persetujuan pelanggan terhadap rencana pilot. Bekerja pada rencana pilot diawali sejak tahap perencanaan sehingga channel komunikasi berda pada tempatnya dan partisipan telah dipersiapkan untuk kesiapan penerapan proyek pilot. Perencanaan pilot sebaiknya meliputi hal-hal berikut :
   a. Jangkauan dan tujuan.
   b. Pengguna yang berpartisipasi, lokasi dan informasi kontak.
   c. Rencana pelatihan untuk pengguna.
   d. Rencana dukungan pilot.
   e. Rencana komunikasi untuk pilot.
   f. Memahami resiko-resiko dan rencana-rencana yang mungkin.
   g. Rencana penarikan solusi.
   h. Rencana penjadwalan dan pelaksanaan pilot.



Gambar 34. Alur Pelaksanaan Pilot.

Setelah rangkain pengujian pilot, tim menyiapkan laporan detail setiap aktivitas sebagai hasil pelaksanaan proyek pilot dan bagaimana informasi-informasi atau isu-isu dapat dipecahkan. Hasil dari proyek pilot meliputi :

- Informasi resiko-resiko tambahan.
- Identifikasi pertanyaan dan jawaban untuk kebutuhan pelatihan.
- Identifikasi kesalahan pemakai untuk pelatihan dan untuk dokumentasi.
- Kemampuan untuk mengamankan dan dukungan untuk pengguna.

- Dokumentasi dan perhatian terhadap isu-isu.
- Memperbaharui dokumentasi menjadi bagian terpisah.
- Menetapkan dimana semua kriteria sukses telah beremu.

Penyelesaian tahap penstabilan memerlukan proses persetujuan, tim membutuhkan dokumentasi hasil terhadap tugas-tugas yang telah dilakukan pada tahap ini untuk mengirimkan proyek ke manajemen untuk persetujuan. Fase penstabilan mencapai kulminasi pada *release readines approved milestone. Milestone* ini terjadi ketika tim telah mengalamatkan semua isu-isu dan merelease solusi dan membuatnya tersedia untuk penyebaran secara keseluruhan. Pada milestone ini merupakan kesempatan bagi pelanggan dan pemakai, personel operasi dan dukungan, stakeholder. Setelah tahap penstabilan lengkap, tim secara formal haruslah setuju bahwa proyek telah mencapai *milestone release readiness*.

10.3. Tahap Penyebaran MSF.

Setelah berakhirnya tahap penstabilan maka solusi telah siap untuk disebarkan. Selama tahap penyebaran tim memasang dan meyebarkan solusi dan komponen, mestabilisasi penyebaran, memindahkan proyek ke aktivitas operasional dan dukungan dan memperoleh persetujuan akhir dari pelanggan. Setelah penyebaran tim melaksanakan review dan mengadakan survey kepuasan pelanggan. Pada tahap ini peran masing-masing anggota tim dapat dibedakan sebagai berikut :

Role	Responsibility
Product management	Umpan balik pelanggan, pengujian, sign-off

Role	Responsibility
Program management	Perbandingan solusi, manajemen penstabilan
Development	Resolusi permasalahan, dukungan eskalasi
User experience	Melatih, manajemen penjadwalan latihan
Testing	Pengujian unjuk kerja, identifikasi masalah, definisi, resolusi, dan pelaporan
Release management	Manajemen Penyebaran, persetujuan terhadap perubahan.

Penyebaran solusi menyesuaikan skenario, misalnya skenario aplikasi web dan layanan, aplikasi *client-server*, aplikasi terpaketkan, infrastruktur interprise, dan aplikasi mobile. Masing-masing memiliki karakteristik penyebaran yang berbeda-beda. Aktivitas penyebaran mempertimbangkan pula kebutuhan atau persyaratan *hardware* dan *operating sistem*. Dalam skenario meliputi *Server Interprise*, *Datacenter* (internet, departemen, global), *web service* dan *client* (*desktop/mobile*).

10.4. Penyebaran Ke Lingkungan Nyata.

Untuk melaksanakan penyebaran haruslah telah disiapkan infrastruktur fisikal, sistem software, software aplikasi yang telah teruji, terpasang dan telah dikonfigurasi pada lingkungan yang akan digunakan. Tim akan memperbaharui dokumentasi-dokumentasi seperti :

1. Diagram penyebaran yang telah dibuat pada tahap perencanaan.
2. Rencana pengujian yang meliputi area yang dijangkau dalam penggunaan nyata. Tim memperbaharui rencana pengujian yang telah dibuat sebelumnya.

3. Rencana pengamanan dimana tim menggunakanya untuk memastikan semua personel menyadari standar keamanan dan aturan proyek.
4. Rencana pembuatan cadangan untuk prefentif kehilangan data, tim akan memperbaharui rencana pembuatan cadangan data.
5. Rencana untuk analisa unjuk kerja sistem dan penggunaanya. Tim akan melakukan perawatan.
6. Rencana untuk penanganan loging (pencatatan) dan tugas-tugas administratif lainya.
7. Rencana pemulihan dari bencana. Tim akan meninjau ulang dokumen, memvalidasi dan memperbaharui konten.
8. Rencana kemungkinan bisnis. Tim akan meninjau ulang, memvalidasi dan memperbaharui konten.
9. Informasi pelatihan. Tim haruslah memastikan dukungan personel yang akan memelihara dan memperbaharui solusi yang telah di diajarkan dengan baik.

Untuk efisiensi penyebaran solusi, sangatlah penting mengelompokkan komponen didalam komponen utama dan spesifik komponen. Komponen utama adalah komponen yang dialokasikan pada lokasi utama yang memungkinkan pelaksanaan operasi antar solusi secara keseluruhan. Komponen-komponen utama terdiri dari :

1. Domain Controler
2. Network Router
3. Database Server
4. Mail Router
5. Remote access server

Komponen spesifik di alokasikan pada lokasi tertentu dan memungkinkan akses dan penggunaan solusi yang disebut site-specific komponen. Instalasi komponen ini biasanya setelah instalasi komponen utama dilakukan. Komponen spesifik tersebut misalnya :

1. Local Router
2. File Print Server
3. Client Application seperti Ms Office

Dalam mengintalasi Komponen-komponen utama dapat di pasang atau di sebarkan dengan dua metode yaitu :

1. Serial, semua komponen utama di sebarkan sebelum penyebaran site, metode ini beresiko lebih kecil dan sesuai dengan lingkungan penyebaran yang kecil
2. Parallel, komponen utama disebarkan sebagai suatu kebutuhan paralel untuk mendukung setiap penyebaran site.

DAFTAR PUSTAKA

[1] Preesman, Roger S. 1994. 'Software Engineering : A Practitioner's Approach'. Third Ed., McGraw-Hill International.

[2] Sommerville, Ian., 2001, 'Software Engineering ', Six Edition, Addison Wesley.

[3] _____,2003, Self Paced Training Kit, 'Analyzing Requirement and Defining Microsoft.NET Solution Architecture, Microsoft Press.

[4] _____, Spiral Model Image, http://www.maxwideman.com/papers/linearity/spiral.htm