YAYASAN PRIMA AGUS TEKNIK

Dasar Konstruksi

RPL

(Rekayasa Perangkat Lunak)

Dr. Budi Raharjo, S.Kom, M.Kom, MM.

Dasar Konstruksi

RPL

(Rekayasa Perangkat Lunak)

Dr. Budi Raharjo, S.Kom, M.Kom, MM.

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 978-623-8642-50-2 (PDF)

Dasar Konstruksi
R P L (Rekayasa Perangkat Lunak)

Penulis :
Dr. Budi Raharjo, S.Kom., M.Kom., MM.

ISBN : 978-623-8642-50-2

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

Penyunting :
Dr. Joseph Teguh Santoso, M.Kom.

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

Penebit :
Yayasan Prima Agus Teknik

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

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

KATA PENGANTAR

Puji syukur kami panjatkan kepada Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya, yang telah memberikan kesehatan dan kesempatan kepada kami untuk menyusun karya ini. Dalam kesempatan ini, kami hadir dengan sebuah pembahasan mengenai *Kontruksi RPL* (Rekayasa Perangkat Lunak), yang diharapkan dapat memberikan pemahaman lebih dalam mengenai proses dan tahapan dalam pembuatan perangkat lunak yang berkualitas.

Rekayasa Perangkat Lunak atau *Software Engineering* (RPL) adalah bidang yang mencakup berbagai prinsip, teknik, dan metode yang digunakan dalam merancang, mengembangkan, dan memelihara perangkat lunak. Konsep ini sangat penting dalam dunia teknologi informasi, di mana perkembangan perangkat lunak menjadi tulang punggung hampir semua sektor kehidupan modern.

Kontruksi RPL sendiri merujuk pada proses pembangunan atau pembangunan perangkat lunak yang melibatkan berbagai tahapan mulai dari perencanaan, analisis, desain, pengembangan, pengujian, hingga pemeliharaan. Proses ini tidak hanya mencakup aspek teknis, tetapi juga memperhatikan aspek manajerial dan kebutuhan pengguna untuk menghasilkan perangkat lunak yang dapat diandalkan dan sesuai dengan harapan pengguna.

Dalam penyusunan karya ini, kami akan mengupas secara singkat namun jelas mengenai tahapan-tahapan penting dalam kontruksi RPL, seperti perencanaan dan analisis kebutuhan, desain sistem, implementasi kode, pengujian perangkat lunak, serta pemeliharaan dan pembaruan. Setiap tahapan ini memiliki peran yang sangat krusial dalam menghasilkan produk perangkat lunak yang berkualitas, aman, dan dapat diandalkan.

Bab pertama dalam buku ini memperkenalkan konsep dasar teknologi, mencakup definisi teknologi dan perkembangan domain teknologi dari masa ke masa. Dalam bab ini juga dibahas mengenai perbedaan teknologi digital dibandingkan dengan teknologi tradisional, serta bagaimana pentingnya pola pikir dalam beradaptasi dengan perubahan tersebut. Penekanan diberikan pada bagaimana pola pikir lean dapat mengubah sektor manufaktur dan konstruksi dengan fokus pada efisiensi dan pengurangan pemborosan. Bab ini mengakhiri pembahasan dengan penjelasan mengenai pasokan teknologi dan bagaimana perubahan teknologi yang cepat mempengaruhi berbagai industri.

Bab kedua membahas perangkat lunak secara rinci, dimulai dengan prinsip dasar perangkat lunak dan bagaimana perangkat lunak berfungsi sebagai mesin. Pembahasan dilanjutkan dengan anatomi perangkat lunak, yang melibatkan struktur dan komponen utama dalam perangkat lunak. Selain itu, bab ini mengulas antarmuka pengguna (UI) dan pengalaman pengguna (UX), serta pentingnya kedua aspek ini dalam menciptakan perangkat lunak yang efektif. Uji coba perangkat lunak dan pemahaman tentang sistem operasi juga menjadi bagian dari pembahasan untuk memahami cara perangkat lunak beroperasi dalam berbagai konteks.

Bab ketiga mengupas tentang jaringan perangkat lunak, menjelaskan berbagai jenis jaringan yang digunakan dalam perangkat lunak serta bagaimana API (antarmuka

pemrograman aplikasi) memfasilitasi komunikasi antar sistem. Pembahasan ini juga menekankan pentingnya memeriksa data secara intuitif dan bagaimana perangkat lunak membentuk suatu sistem yang saling terhubung dan dapat berinteraksi satu sama lain dalam ekosistem teknologi yang lebih besar. Bab keempat fokus pada perangkat lunak konstruksi, dimulai dengan sejarah awal penggunaan perangkat lunak di lapangan konstruksi. Bab ini menjelaskan masalah yang sering dihadapi oleh para profesional konstruksi dalam menggunakan perangkat lunak di lapangan dan memberikan gambaran tentang berbagai jenis perangkat lunak yang digunakan dalam manajemen proyek konstruksi. Selain itu, dibahas juga perangkat lunak CM (Construction Management) sebagai platform utama, serta perangkat lunak lain yang digunakan di luar konteks CM.

Bab kelima mengulas konstruksi industri, menjelaskan berbagai jenis konstruksi yang ada dan bagaimana teknologi informasi (IT) berperan dalam proses konstruksi. Pembahasan ini berfokus pada alat dan proses Information Construction, serta bagaimana teknologi digital dapat mengoptimalkan manajemen dan efisiensi dalam proyek-proyek konstruksi industri. Bab keenam memperkenalkan pembelajaran mesin (machine learning) dan kecerdasan buatan (AI), memberikan pemahaman dasar tentang cara keduanya bekerja dan aplikasinya dalam berbagai bidang. Bab ini juga membahas jenis-jenis pembelajaran mesin, seperti pembelajaran tanpa pengawasan dan pembelajaran mendalam, serta bagaimana AI telah berkembang dalam beberapa tahun terakhir, dengan contoh-contoh penerapannya seperti dalam ImageNet.

Bab ketujuh membahas penerapan kecerdasan buatan (AI) dalam kehidupan nyata, dengan fokus pada pengumpulan data dan cara membangun AI sendiri. Pembaca juga diajak untuk memahami berbagai pendekatan dalam menggunakan API untuk mengintegrasikan AI ke dalam aplikasi, serta bagaimana memilih solusi AI yang sudah ada di pasar. Bab kedelapan mengulas alat-alat teknologi masa depan, seperti realitas virtual (VR), realitas tertambah (AR), pemindaian 3D, serta teknologi IoT (Internet of Things) dan BIM (Building Information Modeling). Pembahasan ini menjelaskan potensi dan aplikasi teknologi-teknologi tersebut dalam industri konstruksi dan berbagai sektor lainnya, serta bagaimana teknologi masa depan ini akan mengubah cara kita bekerja.

Bab kesembilan membahas inovasi dalam teknologi, termasuk inovasi produk, inovasi internal dalam perusahaan, serta bagaimana inovasi dapat merubah budaya organisasi. Pembahasan ini juga menyentuh perbedaan antara inovasi dan kreativitas, serta pentingnya pemikiran desain dalam menciptakan solusi inovatif dalam teknologi dan bisnis. Bab kesepuluh mengulas desain konseptual dalam pengembangan teknologi, menjelaskan tahapan-tahapan desain konseptual dari awal hingga tahap akhir. Pembahasan ini mencakup desain logikal, identifikasi objek bisnis, dan relasi antar objek dalam sistem. Selain itu, pentingnya dokumentasi dan desain fisikal juga dibahas sebagai bagian dari perencanaan dan pelaksanaan desain konseptual.

Bab kesebelas fokus pada desain lapisan data, terutama dalam hal penyimpanan data dan pengelolaan data yang efisien. Pembahasan ini mengulas komponen-komponen penting dalam desain penyimpanan data dan layanan data yang mendukung kinerja aplikasi dan sistem yang lebih besar. Bab kedua belas membahas desain lapisan keamanan, yaitu bagaimana

perangkat lunak dan sistem harus dirancang dengan mempertimbangkan aspek keamanan. Pembahasan ini mencakup mekanisme dan praktik keamanan yang diterapkan dalam pengembangan perangkat lunak, serta bagaimana merencanakan keamanan aplikasi secara efektif. Bab terakhir, bab ketiga belas, mengulas pola pikir konstruksi digital, yang mengedepankan teknologi digital dalam setiap aspek pekerjaan konstruksi. Pembahasan ini menjelaskan pentingnya adopsi teknologi dalam konstruksi serta keterampilan digital yang dibutuhkan oleh para profesional di industri ini. Selain itu, bab ini juga membahas tentang teknologi pembelajaran yang dapat digunakan untuk meningkatkan keterampilan digital dalam dunia konstruksi.

Setiap bab dalam buku ini memberikan wawasan yang mendalam tentang penerapan teknologi dalam berbagai aspek kehidupan, terutama dalam bidang konstruksi, perangkat lunak, dan kecerdasan buatan, serta bagaimana teknologi ini terus berkembang untuk membentuk masa depan industri dan masyarakat.

Semoga buku ini dapat memberikan wawasan yang bermanfaat bagi pembaca yang ingin memahami lebih dalam tentang bagaimana perangkat lunak dibangun secara sistematis dan terstruktur. Penulis berharap agar pembahasan ini juga dapat memberikan inspirasi bagi mereka yang tertarik untuk mengembangkan karir di bidang Rekayasa Perangkat Lunak.

Semarang, November 2024
Penulis

Dr. Budi Raharjo, S.Kom., M.Kom., MM.

DAFTAR ISI

BAB 1
PENDAHULUAN

Cara Anda berpikir tentang dunia memengaruhi apa yang dapat Anda lakukan di dunia.

Dengan berpikir secara berbeda, Anda dapat melakukan hal-hal yang berbeda. Buku-buku seperti ini memperluas cara Anda berpikir, dan karenanya akan memperluas apa yang dapat Anda lakukan – bukan karena pelajaran tentang "bagaimana caranya" yang cepat ketinggalan zaman, tetapi karena kerangka kerja yang kuat untuk melihat semua yang Anda lakukan sebagai semacam teknologi, dan melihat teknologi baru bukan sebagai sesuatu yang terpisah dari apa yang Anda lakukan, tetapi hanya sebagai alat baru dalam perangkat yang diperluas.

Ini adalah buku tentang teknologi yang digunakan dalam konstruksi. "Teknologi" adalah salah satu kata yang digunakan secara berbeda oleh orang yang berbeda, yang membuatnya sulit untuk dibahas. Untuk dapat berpikir dengan jelas, secara berbeda, kita memerlukan definisi konkret tentang apa arti kata-kata seperti "teknologi". Faktanya, poin pertama yang saya ingin Anda setujui, terima, dan pahami adalah bahwa Anda tidak dapat berpikir dengan jelas dengan konsep yang tidak jelas, dan teknologi akan memperkenalkan Anda pada banyak konsep yang tidak jelas bagi Anda pada awalnya. Dalam buku ini, kita akan berhenti sejenak dan mendefinisikan sebanyak mungkin istilah baru.

Konstruksi adalah industri yang terdiri dari berbagai keterampilan dan praktik yang diajarkan dengan cara menunjukkan dan berbicara, jadi budayanya tidak selalu menanyakan langsung kepada orang yang tidak tahu apa yang mereka bicarakan. Mungkin ada rasa tidak nyaman untuk bertanya, karena pada titik tertentu teknologi, khususnya perangkat lunak, telah membuat semua orang merasa bodoh.

Bacalah buku ini dan saya jamin hal itu akan lebih jarang terjadi. Namun, intinya adalah merasa yakin bahwa bukan ketidaktahuan Anda tentang konsep baru apa pun yang sedang dibahas, tetapi kegagalan vendor atau penyaji untuk memastikan adanya pemahaman bersama.

Dalam hal produk dan proses teknologi, selalu menjadi tugas penyedia untuk memastikan Anda jelas – minta mereka menaatinya.

1.1 APA ITU TEKNOLOGI

Jadi, mari biasakan diri untuk membuat definisi yang jelas dengan membuat definisi untuk teknologi:

Teknologi adalah penerapan beberapa efek, biasanya ilmiah, untuk menyelesaikan pekerjaan.

Kata "teknologi" juga dapat digunakan untuk dua tingkat makna lainnya:

1. Kumpulan hal yang bekerja dengan cara yang sama, seperti teknologi konstruksi.

2. Seluruh kelas upaya manusia yang menciptakan alat untuk budaya tertentu, seperti teknologi digital.

Kita akan fokus pada makna pertama. Penting untuk berpikir pada level ini terlebih dahulu, karena Anda akan berhadapan dengan produk-produk spesifik, bukan pengelompokan besar atau kelas-kelas abstrak produk.

Ketika berhadapan dengan produk teknologi baru, seperti perangkat lunak konstruksi, kita dapat dikejutkan oleh apa yang tidak kita ketahui, dikejutkan oleh betapa berbedanya hal itu dengan cara-cara yang telah kita lakukan di masa lalu. Namun, teknologi tidak muncul begitu saja. Agar bermanfaat, mesin, proses, atau perangkat lunak baru akan dikembangkan sehingga Anda dapat melakukan sesuatu yang sudah Anda lakukan, lebih cepat, lebih aman, atau lebih murah.

Memahami Dasar Teknologi

Teknologi apa pun didasarkan pada beberapa efek yang mendasarinya, beberapa kesadaran bahwa alam, atau sifat manusia, bekerja dengan cara tertentu. Ada beberapa efek, atau fenomena, yang membuat teknologi tersebut bekerja. Jadi, kita membangun suatu proses, atau alat, atau mesin, yang memanfaatkan efek ini untuk membuat pekerjaan manusia menjadi lebih baik dengan cara tertentu. Sering kali, teknologi ini membuat hal-hal yang tidak mungkin menjadi mungkin.

Misalnya, pikirkan tentang palu. Kita tidak menganggapnya sebagai teknologi, tetapi memang demikian. Berikut ini adalah beberapa efek di dunia yang dieksploitasi oleh palu genggam modern:

1. Setiap gaya menciptakan gaya yang sama dan berlawanan (hukum ketiga Newton, yang sama yang digunakan dalam roket)
2. Baja itu keras
3. Baja karbon tinggi yang digulung dingin sangat keras
4. Logam lebih keras daripada kayu atau gipsum
5. Ujung bandul lebih cepat daripada gagangnya
6. Gaya yang diberikan pada area tertentu akan berlipat ganda ketika dipindahkan ke area yang lebih kecil

Semua itu ada dalam palu sederhana. Bayangkan apa yang dilakukan palu: palu menggunakan gerakan dari lengan manusia untuk mentransfer gaya dari satu objek baja, kepala palu, ke objek baja lain, paku. Gaya ini kemudian mendorong paku menembus material apa pun yang sedang dikerjakan.

Mari kita luangkan waktu sejenak dan pikirkan tentang apa yang Anda lakukan, sepanjang hari. Baik itu memasang saluran listrik, mengelola tim kontraktor mekanik, mengelola lokasi kerja sebagai pengawas, atau mengelola seluruh pekerjaan sebagai manajer proyek – semua yang Anda lakukan berhasil karena beberapa efek di dunia. Beberapa efek tersebut sangat manusiawi, seperti ego, kesombongan, dan keinginan untuk menciptakan sesuatu yang nyata di dunia. Dan Anda belajar melalui karier Anda untuk menggunakan efek tersebut untuk memotivasi, mengelola, atau sekadar mengarahkan orang lain. Misalnya, Anda belajar untuk sering memeriksa orang lain karena Anda tahu bahwa akuntabilitas membuat

orang lebih fokus pada pekerjaan – efek yang Anda manfaatkan untuk menyelesaikan pekerjaan. Manajemen adalah teknologi yang sama rumitnya dengan kecerdasan buatan – faktanya, sebagai seseorang yang telah melakukan keduanya, saya dapat memberi tahu Anda bahwa manajemen bisa jadi lebih sulit karena merupakan tindakan penyeimbangan yang tidak pernah berakhir. Manajemen sebagai sebuah praktik telah berevolusi dari waktu ke waktu untuk menggunakan metode yang berbeda, masing-masing menggunakan efek yang berbeda di dunia – kita dulu hanya mengandalkan hierarki dan kekuasaan, yang mengandalkan rasa takut kehilangan pekerjaan. Namun, kami menyadari bahwa hal itu menghambat arus informasi penting dan menyebabkan pekerja tidak terlibat. Jadi, kami telah mengganti efek yang mendasarinya menjadi perasaan terlibat dan berprestasi, yang menjadi fokus Lean Construction. Mengubah efek yang menjadi dasar suatu teknologi bisa sangat hebat.

Kembali ke contoh palu kita, apa yang terjadi jika kita memisahkan pekerjaan yang harus dilakukan, yaitu menancapkan paku ke dinding, dari cara penyelesaiannya, yaitu gaya sentrifugal dari sepotong baja yang diayunkan dan mengenai kepala paku yang diam? Bagaimana jika kita meletakkan paku di dalam tabung yang terhubung ke udara bertekanan, dan "menembakkan" paku ke dinding? Kami telah mengubah efek yang dieksploitasi untuk menyelesaikan pekerjaan yang sama, dari ayunan lengan manusia menjadi pelepasan tekanan udara mekanis, dan dalam prosesnya telah secara dramatis meningkatkan efisiensi pekerja pemaku kami, secara signifikan meningkatkan kemampuan mereka untuk terus memaku tanpa kelelahan atau cedera lengan, dan mudah-mudahan terhindar dari kuku jempol mereka.

1.2 DOMAIN TEKNOLOGI

Ketika kita menganggap teknologi sebagai rangkaian efek-efek ini yang digunakan, kita dapat lebih mudah memahami cara mengintegrasikan bagian-bagian baru. Teknologi hanyalah alat yang merupakan perpanjangan dari kekuatan dan tindakan manusia, tidak lebih, tidak kurang. Dengan mengingat bahwa teknologi apa pun ada untuk memperluas kekuatan dan tindakan Anda, Anda dapat meletakkannya dalam konteks yang membantu menilai seberapa bagus teknologi itu, dan memahami mengapa teknologi baru mungkin lebih baik daripada yang lama – teknologi tersebut menggunakan efek yang lebih baru dan lebih baik untuk membantu Anda. Faktanya, lebih sering terjadi bahwa alat yang Anda gunakan mengeksploitasi sekumpulan efek untuk bekerja, dan bagian yang paling Anda pedulikan, efek yang dieksploitasi yang paling penting bagi Anda adalah efek manusia dan organisasi. Dan dengan memahami semua teknologi sebagai bagian dari rantai yang Anda ikuti, Anda dapat lebih memahami pencipta teknologi tersebut tentang cara mereka mengembangkan dan menghasilkan produk nyata. Yang terjadi dalam konstruksi sekarang adalah bahwa bagian dari rantai teknologi, mulai dari keterampilan Anda hingga alat yang Anda gunakan, sedang ditambahkan dan diubah. Memahami teknologi modern bukan sebagai kelas benda yang terpisah, tetapi hanya yang terbaru dalam serangkaian "modul" yang dapat ditukar ke dalam rantai yang sudah ada adalah perspektif yang jauh lebih baik, karena hal itu membuat teknologi tetap berada di bawah kategori "alat" – bukan sesuatu yang asing.

Teknologi itu sendiri selalu merupakan kumpulan dari teknologi lainnya. Bahkan sesuatu yang sederhana seperti palu merupakan hasil dari plastik, metalurgi, pengerjaan logam, desain pabrik, otomatisasi, ergonomi, dan bahkan pengemasan. Mengetahui bahwa teknologi itu sendiri terdiri dari teknologi tingkat rendah yang saling terkait, memudahkan kita untuk memahami bagaimana peralatan kita terus menjadi sedikit lebih baik, lebih murah, dan lebih aman setiap tahun. Misalnya, Milwaukee Tool telah membuat peralatan seperti kunci torsi selama bertahun-tahun, dan seiring waktu mereka telah mengganti beberapa kontrol yang mekanis, seperti tuas langsung antara pemicu dan motor listrik, dengan elektronik yang memberi pengguna penyetelan operasional yang lebih baik.

Milwaukee baru-baru ini menambahkan fitur baru, "One-Key System," yang melacak semua peralatan di lokasi kerja secara nirkabel. Anda dapat melihat ini sebagai fitur lain, yang sebenarnya cukup keren, atau Anda dapat melihatnya sebagai Milwaukee yang mengganti kertas dan pensil, atau mungkin teknologi spreadsheet yang mengandalkan observasi, memori, dan melakukan putaran untuk tetap mengikuti perkembangan, dengan alat pelaporan inventaris otomatis dan waktu nyata. Rantai teknologi Anda tidak berubah, tetapi alat yang Anda gunakan untuk melakukan sebagiannya telah berubah.

Memikirkan pekerjaan Anda sebagai rantai alat, dari keterampilan Anda sendiri hingga alat eksternal yang Anda gunakan memberi Anda, profesional yang menggunakan teknologi ini, kekuatan untuk berpikir kreatif tentang apa yang Anda gunakan, menggabungkan produk dan teknologi dengan cara baru, dan menuntut penyedia produk teknologi untuk terus bekerja hingga produk tersebut melakukan apa yang Anda butuhkan.

Ide mengganti satu jenis teknologi dengan yang lain sering disebut sebagai mengubah "ranah" teknologi yang digunakan. Dalam contoh palu, kami mengubah ranah gaya manual menjadi ranah gaya pneumatik. Dalam contoh alat Milwaukee, kami mengubah ranah inspeksi dan pelaporan manual menjadi ranah pembaruan otomatis nirkabel. "Penggantian domain" atau pertukaran satu domain teknologi dengan domain lain dalam aktivitas manusia ini telah terjadi sejak awal waktu. Domain-domain baru ini cenderung muncul secara bergelombang, dan saat muncul, ada proses yang tak terelakkan untuk memadukan pengalaman dan penilaian profesional industri dengan proses dan keterampilan baru yang dibawa oleh teknologi baru ini. Itulah yang sedang kita alami sebagai sebuah industri, mengganti teknik dan teknologi era fisik dan industri dengan domain digital.

Dari contoh-contoh di atas, pekerjaan manusia digantikan oleh mesin. Ketakutan berlimpah dalam perdagangan khusus, dan kontraktor secara umum, bahwa teknologi akan menyebabkan orang kehilangan pekerjaan mereka. Kami akan membahas ini secara sangat spesifik dalam bab AI dan Konstruksi Industri, tetapi untuk saat ini, ingatlah dua hal: yang pertama adalah bahwa mesin dan perangkat lunak menggantikan keterampilan dari yang paling bawah, dimulai dengan tugas-tugas yang paling tidak memerlukan pikiran. Beberapa dari tugas-tugas tersebut masih memerlukan keterampilan, seperti menggunakan palu dengan baik, tetapi keterampilan tersebut merupakan bagian kecil dari apa yang menjadikan seorang tukang kayu sebagai seorang profesional. Hal kedua adalah bahwa yang menjadikan seorang tukang kayu, atau tukang pipa, atau kontraktor mekanik sebagai seorang profesional

adalah kemampuan untuk memecahkan masalah, dan untuk menghasilkan solusi atas kendala-kendala rumit yang melibatkan jadwal, ketentuan kontrak, dinamika tim, dan banyak hal lainnya.

Domain Versus Produk

Tampaknya perubahan dalam teknologi tidak dapat dihindari, dan ketika dilihat sebagai kumpulan solusi dan produk, ada beberapa kebenaran dalam hal itu. Sains akan terus menghasilkan efek dan wawasan baru yang dapat kita manfaatkan. Perusahaan, yang didorong oleh perancang untuk saling mengalahkan, akan terus menyempurnakan cara untuk memanfaatkan efek tersebut.

Namun, tidak ada yang tak terelakkan dari teknologi apa pun. Kita yang berkecimpung di industri ini benar-benar dapat memengaruhi produk apa yang ada di luar sana, dan yang lebih penting, bagaimana produk tersebut bekerja untuk kita, dan bersama kita.

Contoh kontras antara jenis teknologi yang mungkin akan terjadi apa pun yang terjadi, dan produk tertentu yang sangat mungkin tidak terelakkan adalah pertarungan VHS/Beta pada tahun 1980-an. Dimulai pada tahun 1970-an, film dan TV mengalami revolusi penyimpanan digital, yang menjadi kaset video. Ada dua pilihan produk: Betamax dan VHS. Beta adalah video berkualitas lebih tinggi, karena pengembangnya, Sony, berasumsi konsumen ingin menikmati film dan TV mereka dengan kualitas visual setinggi mungkin, yang mereka capai dengan membatasi pemutaran hingga satu jam. Sebaliknya, VHS Panasonic memberikan kualitas visual yang lebih rendah tetapi berdurasi hingga 2 jam saat pertama kali diperkenalkan. Karena sebagian besar film berdurasi lebih dari satu jam, Beta tidak begitu cocok dengan pasar, dan akhirnya kalah bersaing dengan VHS, yang menjadi standar selama sekitar satu dekade.

Dua hal yang berperan – standar dan produk. Dalam kasus ini, pasar kaset video konsumen didominasi oleh standar VHS, dan produk-produk tertentu semuanya adalah VHS.

Sebelum ini, satu-satunya cara untuk menonton film di rumah adalah ketika jaringan penyiaran memilih untuk menayangkan film tersebut, yang jarang mereka lakukan dan dalam bentuk yang sangat diedit dan ramah anak. Pasar video rumahan dengan demikian "diubah domainnya" sehingga cara konsumen menonton film di rumah berubah dari model yang berpusat pada penyiaran, menjadi model yang berpusat pada kaset video. Domain baru akan terjadi, tetapi produk-produk tertentu akan menang atau kalah berdasarkan seberapa baik mereka cocok dengan pasar. Dan Anda, pembaca yang budiman, adalah pasarnya. Anda memutuskan siapa yang menang atau kalah.

Hal ini lebih penting daripada yang mungkin terlihat, karena ketika domain baru muncul, sering kali ada banyak pilihan untuk sementara waktu. Beberapa di antaranya gagal, beberapa dibeli atau digabungkan dengan yang lain, dan beberapa akan menang. Semua orang tahu tentang Facebook, dan beberapa mungkin ingat MySpace. Namun, apakah Anda ingat Friendster dan sekitar 40 jejaring sosial lain yang muncul pada pertengahan tahun 2000-an?

Itulah yang terjadi sekarang dengan perangkat lunak manajemen proyek konstruksi, di mana mulai sekitar tahun 2015, semakin banyak perusahaan yang mulai mendigitalkan alur

kerja konstruksi – mengubah domain teknologi dari kertas dan Microsoft Excel, menjadi platform terpadu yang menangani berbagai bagian dari proses konstruksi, atau semuanya. Pada waktunya, akan ada pemenang dan pecundang – mungkin bukan monopoli yang kita lihat di pasar konsumen, tetapi penawaran produk yang lebih sedikit.

Pengubahan domain sering kali dapat berasal dari industri lain. Ambil contoh keamanan gedung. Kaset video memungkinkan kemampuan yang sama sekali baru untuk menangkap dan menyimpan video dari kamera di gedung, kamera yang telah diubah dari film menjadi digital dalam rentang waktu sekitar tahun 1980-an yang sama. Namun pada akhir tahun 1990-an, konten dari berbagai jenis, mulai dari musik hingga video dan gambar, semuanya mulai diubah ke dalam format digital. Format ini awalnya berupa CD, tetapi kemudian berubah menjadi video MPEG dan audio MP3. Sekitar tahun 2000, pasar konsumen mulai memasarkan pemutar untuk format digital ini, yang menurunkan biaya penyimpanan digital untuk keamanan, yang mengarah ke sistem masa kini yang sepenuhnya digital. Faktanya, sistem seperti keamanan gedung telah mengalami serangkaian perubahan domain teknologi, dari kaset video, ke CD, ke hard drive, hingga cloud baru-baru ini.

Sering kali terjadi bahwa perkembangan di pasar lain, khususnya pasar konsumen, menciptakan tekanan untuk melakukan perubahan di pasar lain, karena teknologi menjadi murah dan familiar bagi pengguna.

Perubahan model ini, redomaining, akan terjadi apa pun yang terjadi. Namun, cara spesifik terjadinya perubahan tersebut bukanlah hal yang tak terelakkan. Ini adalah pelajaran penting bagi teknologi dalam konstruksi: tekanan kemajuan teknologi berarti bahwa proses konstruksi akan terus mengalami digitalisasi, akan terus menyerap dan mengintegrasikan teknologi baru, tetapi perusahaan atau produk apa pun dapat berhasil atau gagal. Cara menilai produk-produk ini akan menjadi salah satu hal utama yang dapat diambil dari buku ini, saat kita membahas setiap bidang teknologi yang akan Anda temui.

1.3 APA YANG BERBEDA TENTANG TEKNOLOGI DIGITAL

Tidak seorang pun perlu menulis Buku Pegangan Teknologi Konstruksi ketika teknologi yang digunakan terbatas pada perdagangan individu, dan melibatkan peralatan dan mesin yang sebagian besar memanfaatkan efek fisik. Belajar menggunakan pistol paku bukanlah lompatan besar dari palu; belajar menggunakan peralatan listrik yang semakin kuat dan canggih biasanya merupakan proses evolusi di mana fitur-fiturnya terus menjadi lebih baik. Dalam hal ini, pekerja dapat melihat cara kerja teknologi, dapat memahami secara intuitif cara menggunakan teknologi, meskipun mungkin butuh waktu bertahun-tahun untuk menguasainya secara keseluruhan.

Sebaliknya, teknologi digital bekerja tanpa terlihat, dengan cara nonfisik yang tidak dapat langsung dipahami manusia, menggunakan kontrol yang tidak "alami" seperti gagang palu. Teknologi lama memanfaatkan fisika, yang secara alami dipahami manusia.

Teknologi digital memanfaatkan fenomena elektronik, yang sangat kecil sehingga tidak dapat kita lihat. Teknologi digital juga membangun lapisan antarmuka yang dirancang manusia yang sama sekali tidak harus bergantung pada intuisi atau gerakan alami, semuanya

sepenuhnya merupakan penemuan pengembang produk. Dan itu berarti intuisi dan pengalaman yang bekerja dengan sangat baik dengan teknologi berbasis fisika tidak membantu kita dengan teknologi digital, dan itu bisa membuat kita terasing dan kesal. Antarmuka yang dirancang manusia ini memang memiliki logika, dan didasarkan pada rekayasa nyata – sehingga Anda dapat mempelajari logika itu dan menjadi sama nyamannya dengan teknologi digital seperti halnya dengan hal lainnya.

Logika itu, "fisika" teknologi digital itulah yang dibahas dalam buku ini. Kami telah memulainya dari awal, mendefinisikan apa yang kami maksud, dan akan membangun aturan dan kerangka kerja yang memberi Anda pemahaman mendalam tentang apa yang membuat teknologi digital bekerja.

Komunitas Anda

Selama bertahun-tahun, industri Arsitektur, Teknik, dan Konstruksi (AEC) termasuk yang paling lambat mengadopsi teknologi, sebagian karena teknologi yang diperkenalkan sejak awal sering kali tidak disesuaikan dengan kebutuhan industri dan berbagai bagiannya. Dalam satu dekade terakhir, sejumlah upaya komunitas telah muncul untuk memberi industri AEC, khususnya profesional konstruksi, kesempatan untuk belajar tentang teknologi, sekaligus memberi industri teknologi jendela ke salah satu industri tertua dan paling rumit.

Mungkin yang pertama adalah AEC Hackathon, rangkaian acara yang saya bantu mulai pada tahun 2013. Didirikan oleh Damon Hernandez dan Paul Doherty, acara ini dibuat untuk mendobrak batasan antara orang-orang industri AEC dan orang-orang teknologi. Dengan memecahkan masalah nyata bersama-sama, kedua belah pihak saling menghargai. Kami telah menjalankan lebih dari 50 di seluruh dunia, mengubah formatnya menjadi versi daring di dunia pascapandemi. Sekarang setelah semuanya digital, saya anjurkan Anda untuk memeriksanya; kunjungi www.hackaec.com untuk melihat apa yang ada di luar sana, dan temukan orang lain seperti Anda yang sedang menjalani perjalanan digital.

Pada tahun-tahun sejak 2013, perusahaan rintisan dan modal ventura telah menemukan konstruksi, dan telah ada banyak solusi untuk segala hal mulai dari pemindaian 3D hingga laporan harian. Tidak semua ini berasal dari pemahaman yang baik tentang apa yang benar-benar dibutuhkan di lokasi kerja, dan bahkan jika melihat lebih jauh ke belakang, lebih banyak perangkat lunak yang didorong ke lokasi konstruksi berasal dari tempat lain, seperti akuntansi. Kami telah mendengar cerita tentang "kelelahan aplikasi," dan kekhawatiran umum bahwa personel lapangan khususnya tidak cukup berperan dalam proses pengembangan dan adopsi teknologi.

Kita perlu mengubahnya.

Dalam buku ini, saya berbagi keterampilan baru, dan pola pikir baru terhadap teknologi. Apakah Anda sudah menjadi teknolog konstruksi dan telah "minum kool-aid," atau khawatir tentang bagaimana teknologi akan memengaruhi hidup Anda, Anda akan menemukan ide-ide berharga di halaman ini.

Melalui perangkat digital baru ini, Anda akan melihat teknologi secara berbeda, dari cara pembuatannya hingga cara pengemasan dan pengadopsiannya. Anda juga akan melihat bahwa orang-orang yang membuat perangkat lunak dan jenis teknologi lainnya sangat mirip

dengan konstruksi, terutama dalam bidang perdagangan. Perbedaan besarnya adalah keahlian konstruksi ditujukan untuk menempatkan pekerjaan di tempat, sedangkan keahlian teknologi ditujukan untuk "mewujudkannya." Kedua kelompok profesional tersebut sangat bangga dengan apa yang mereka lakukan, dan dengan memahami teknologi hanya sebagai perangkat lain, Anda dapat memadukan kekuatan pola pikir konstruksi dan teknologi.

Sering dikatakan bahwa pelatihan mengajarkan Anda cara bertindak, pendidikan mengajarkan Anda cara berpikir. Buku seperti ini bersifat mendidik – jadi ajukan pertanyaan di sepanjang jalan, dan cobalah untuk melihat berbagai hal dengan pola pikir seorang teknolog.

1.4 POLA PIKIR ITU PENTING

Apa yang kita maksud dengan "pola pikir?"

Kami memulai buku ini dengan sebuah pernyataan: Cara Anda berpikir tentang dunia mengubah apa yang Anda lakukan, dan cara Anda melakukannya.

Itulah pola pikir. Dan janji yang dibuat oleh ide tersebut adalah ini: Ubah pola pikir Anda dan Anda dapat mengubah kemungkinan Anda. Siapa pun yang pernah berolahraga pasti setuju bahwa pola pikir adalah segalanya.

Apa saja perubahan pola pikir dalam pekerjaan atau kehidupan nyata Anda? Pertama-tama, pola pikir mengubah apa yang Anda perhatikan, dan apa yang menurut Anda berharga untuk waktu Anda.

Dalam lingkungan yang kompleks dan bergerak cepat seperti konstruksi, Anda harus memilih apa yang penting dari serangkaian acara, rapat, dan pesan. Anda tidak dapat menemukan apa yang penting jika Anda tidak memiliki gambaran tentang cara kerja dunia – model tentang apa yang menyebabkan apa, dan apa yang penting pada akhirnya.

Konstruksi selalu memiliki perangkat yang sangat banyak, mulai dari mesin las MIG hingga perkakas tangan hingga mesin berat, dan segala hal di antaranya. Sebagian besar peralatan tersebut didasarkan pada proses yang telah berevolusi secara perlahan dan, seperti yang dibahas di atas, didasarkan pada efek yang dapat diamati. Untuk membuat proses tersebut berhasil, pola pikir konstruksi adalah mengandalkan pengalaman masa lalu, memercayai insting Anda, dan kecemasan terus-menerus tentang apa yang mungkin salah yang tidak dapat Anda lihat. Pola pikir inilah yang menjadi alasan mengapa para pengawas, bahkan hampir semua orang, terus-menerus memeriksa lokasi kerja – mereka melihat langsung ke segala hal, hanya memercayai mata mereka sendiri. Pola pikir dan model mental yang membuat seseorang ahli dalam konstruksi secara tradisional tidak sama dengan pola pikir yang akan membuat Anda menguasai kedua bidang yang sudah Anda kuasai, dan perangkat digital yang Anda gunakan sekarang dan akan dapat Anda gunakan di masa mendatang. Tidak ada alasan Anda tidak dapat menggunakan kedua pola pikir tersebut, karena keduanya tidak benar-benar bertentangan. Pola pikir konstruksi digital memisahkan yang fisik dari yang digital, memahami bahwa masing-masing mendukung dan membutuhkan yang lain agar berfungsi. Pahami bahwa beberapa masalah ada dalam benak, dan beberapa

ada dalam analisis. Kedua perangkat tingkat tinggi ini sangat penting, dan keduanya juga saling tumpang tindih.

Masalah Intuitif

Proses dan masalah yang dapat Anda lihat secara langsung akan selalu menjadi hal yang paling utama karena pengalaman dan intuisi. Tidak ada mesin atau perangkat lunak yang dapat dibandingkan dengan kecanggihan yang dibawa oleh seorang profesional berpengalaman ke dalam situasi nyata.

Hal ini berlaku karena beberapa alasan, yang terpenting adalah tidak ada perangkat lunak yang telah diciptakan, termasuk kecerdasan buatan (AI) yang sangat digembar-gemborkan, yang memiliki pemahaman akan konteks. Manusia sangat bergantung pada konteks, dan kita memahami tidak hanya apa yang ada di depan kita, tetapi juga bagaimana hal itu berhubungan dengan apa yang terjadi kemarin, bagaimana hal itu mungkin berhubungan dengan pria dan wanita tertentu di tempat kerja, dan faktor-faktor lainnya. Teknologi digital tidak dapat melakukan ini.

Namun konteks tidak hanya berasal dari apa yang Anda dengar atau lihat, tetapi juga dapat berasal dari teknologi digital – laporan dan analisis gambaran yang lebih besar, yang membantu intuisi Anda melakukan tugasnya.

Masalah Digital

Intuisi bekerja dengan menyederhanakan dunia, dengan aturan praktis mental, sehingga kita dapat menerima semua informasi dan melakukan sesuatu dengan cepat. Kita berevolusi untuk bertindak cepat, untuk melihat masalah dengan segera, itulah sebabnya kita sangat pandai mengelola apa yang dapat kita lihat.

Namun, konstruksi modern melibatkan banyak orang, selama berbulan-bulan bekerja dan di beberapa lokasi. Itu terlalu banyak untuk ditangani oleh otak kita secara langsung, jadi kita menggunakan tumpukan kertas dan banyak rapat untuk menyelesaikan semuanya, atau kita mengubah domain teknologi yang kita gunakan menjadi perangkat lunak digital.

Masalah digital melibatkan banyak data, baik karena cakupan informasi yang besar, atau pengukuran yang sangat teliti yang hanya dapat dilakukan oleh mesin. Misalnya, melacak kejadian dan kemajuan di seluruh perusahaan, atau bahkan pekerjaan, akan mencakup ribuan titik data, selama berminggu-minggu dan berbulan-bulan selama masa pakai proyek. Ini bukanlah hal yang dapat dilakukan oleh otak kita dengan baik, dan di sinilah teknologi digital membantu – dengan apa yang tidak dapat kita alami secara langsung dengan indera kita.

Analisis yang berasal dari perangkat lunak digital dapat menunjukkan kepada kita pola yang tidak akan kita lihat, dan yang paling penting, menciptakan bukti bahwa pola ini ada, sehingga kita dapat mendiskusikannya dengan manajemen dan mulai membuat perbaikan. Namun, seperti halnya analitik yang dapat memberikan konteks bagi intuisi, "pemeriksaan naluri" intuitif pada data dan perangkat lunak sangatlah penting. Tidak ada perangkat lunak yang benar-benar dapat memperbaiki dirinya sendiri ketika memiliki data yang buruk atau masalah tak terduga lainnya. Pengawasan manusia terhadap perangkat lunak, robot, dan teknologi digital lainnya sangatlah penting.

Pergeseran pola pikir kita, kemudian, adalah menerima teknologi digital sebagai alat untuk melihat skala dan cakupan informasi yang tidak dapat dilihat secara langsung. Kita perlu membangun kepercayaan bahwa teknologi ini dapat melakukan apa yang tidak dapat dilakukan manusia sendiri, tetapi juga memahami keterbatasan inherennya.

Kita akan mengembangkan pemahaman yang jelas tentang teknologi apa yang dapat, dan tidak dapat, melengkapi cara Anda melakukan sesuatu. Sama seperti manufaktur saat ini mencakup semua pengalaman, intuisi, dan "naluri" dari manajer berpengalaman dan menerapkannya pada mesin, perangkat lunak, dan analitik yang sangat canggih, teknologi sebagai bagian dari perangkat konstruksi Anda hanya masuk akal jika menambah apa yang sudah Anda kuasai.

Tidak ada yang dapat menggantikan kemampuan pikiran manusia yang hampir ajaib untuk memahami apa yang sedang terjadi dan dengan mudah, mengetahui apa yang harus dilakukan selanjutnya. Ini adalah buku tentang teknologi, tetapi teknologi yang paling penting adalah cara Anda berpikir.

Pola pikir teknologi konstruksi ini akan mengubah cara kita membangun dunia. Mari kita lihat bagaimana, dengan contoh dari manufaktur, khususnya pola pikir Lean.

1.5 BAGAIMANA POLA PIKIR LEAN MENGUBAH MANUFAKTUR

Manufaktur mobil di Jepang sebenarnya adalah asal mula semua "Lean" – berdasarkan inovasi Kiichiro Toyoda dan Taiichi Ohno, bos perusahaan otomotif Toyota pada tahun-tahun setelah Perang Dunia II. Dihadapkan dengan hampir tidak ada modal dalam industri padat modal, pekerja dengan keterampilan terbatas dalam industri padat keterampilan, dan persaingan yang hampir tak terkalahkan dari Detroit di pasar yang kompetitif, Toyoda dan Ohno harus menemukan cara baru dalam memandang bisnis mereka.

Mereka melakukan ini dengan kembali ke prinsip-prinsip awal, dan mempertanyakan segala hal tentang bagaimana mobil diproduksi, untuk melihat bagaimana mereka dapat menciptakan cara baru. Hal pertama yang mereka pertanyakan adalah bagaimana keberhasilan diukur di lantai pabrik – secara tradisional hal ini didorong oleh fakta bahwa mesin pengepres dan perakitan besar (dan masih) sangat mahal. Jadi, tujuan dari sudut pandang finansial adalah selalu memastikan Anda memaksimalkan laba atas investasi (ROI).

Mengikuti pola pikir ROI ini, manajer perusahaan mobil tradisional dan operator lantai bekerja untuk memastikan bahwa setiap mesin digunakan sebanyak mungkin, yang berarti memproduksi sebanyak mungkin komponen yang diproduksi per jam, sehingga biaya mesin dapat dibagi ke seluruh komponen tersebut. Dalam pola pikir yang berpusat pada ROI ini, manajer lebih menghargai efisiensi mesin daripada efisiensi seluruh proses.

Pikirkan apa artinya jika Anda memiliki, katakanlah, lima mesin berturut-turut yang membuat obeng. Yang pertama mengeluarkan kawat, memotongnya, dan meneruskannya. Yang berikutnya membenturkan ujung depan ke bilah datar, dan membenturkan ujung belakang untuk membuat jangkar kecil. Langkah ketiga mencelupkan batang logam ke dalam cetakan plastik untuk membuat pegangan. Langkah keempat mengoleskan cat dan pelapisan, dan langkah kelima memoles dan menyiapkan obeng yang baru dicetak untuk dikemas.

Ingatlah bahwa cara klasik untuk melihat proses ini adalah kita ingin menjalankan setiap mesin semaksimal mungkin, untuk mencapai ROI ajaib kita. Bagaimana jika setiap mesin, setiap bagian dari proses memiliki kecepatan yang berbeda, menghasilkan jumlah output yang berbeda per jam? Bagaimana jika salah satu proses awal tidak menghasilkan komponen berkualitas tinggi, yang kemudian berlanjut ke proses selanjutnya?

Jelas hal ini terjadi sepanjang waktu, dan sebagian besar pabrik pra-Lean menjadi kacau karena pemborosan dan masalah kualitas sebagai akibatnya. Kembali ke contoh obeng kita, apa yang terjadi ketika Anda menjalankan semua mesin dengan kecepatan tinggi, masing-masing memproduksi sebanyak mungkin, adalah Anda mendapatkan banyak batang logam yang tergeletak di sekitar, menunggu langkah pencetakan plastik, atau Anda mendapatkan banyak obeng setengah jadi yang menunggu langkah pemolesan. Itu membuat lantai pabrik menjadi kotor, menghabiskan uang untuk persediaan, dan mengalihkan fokus dari apa yang seharusnya dilakukan pabrik, yaitu membuat barang-barang yang ingin dibayar orang. Telah dilaporkan dalam lebih dari satu penelitian bahwa lebih dari 90% pengawas konstruksi menganggap lokasi kerja mereka tidak efisien – jelas beberapa masalah yang sama yang dihadapi oleh manufaktur pra-Lean terjadi di lokasi kerja kami. Pola pikir pra-Lean adalah berfokus untuk mendapatkan hasil maksimal dari mesin, padahal seharusnya menghasilkan produk sebebas mungkin dari cacat. Mesin-mesin tersebut berharga sesuai dengan harganya, baik Anda mengoperasikannya 100% atau 50%. Itu sudah pasti. Namun, Anda dapat mengontrol berapa banyak inventaris yang telah Anda buat, dan jika Anda berfokus pada keseluruhan proses, Anda akan bertanya apakah setiap mesin benar-benar perlu diatur pada kecepatan 100%, atau mungkin apakah Anda dapat menyeimbangkan berbagai hal sehingga setiap langkah hanya menghasilkan cukup bahan untuk langkah berikutnya. Manajer yang memahami pola pikir lean memahami bahwa tugas pabrik bukanlah untuk memanfaatkan mesin-mesinnya sebaik-baiknya. Tugas pabrik adalah memanfaatkan prosesnya sebaik-baiknya, dari total kumpulan pekerja, material, dan mesinnya. Namun, perubahan yang sesungguhnya adalah dalam peran perencanaan dari atas ke bawah dibandingkan dari bawah ke atas. Anda lihat, begitu pola pikir itu diubah, disadari bahwa pekerja garis depan memahami proses sehari-hari jauh lebih baik daripada manajer, jadi mereka diberi wewenang untuk menghentikan lini dan membuat penyesuaian, menetapkan rencana jangka pendek, dan melakukan perbaikan.

Pekerja yang sama ini dikumpulkan ke dalam kelompok-kelompok kecil, yang disebut "lingkaran kualitas", yang akan sering bertemu dan membicarakan tentang bagaimana bagian-bagian proses mereka dapat ditingkatkan - biasanya bagian ini mencakup beberapa langkah yang berdekatan. Dan akhirnya, banyak data dikumpulkan tentang bagaimana proses secara keseluruhan berjalan sehingga peluang yang ditingkatkan pada tingkat itu dapat dilihat dan keputusan dibuat.

Selama paruh kedua abad kedua puluh, pemikiran Lean membawa Jepang dari negara yang dilanda kemiskinan dan kelelahan menjadi salah satu negara dengan ekonomi terkemuka di dunia.

Ingatlah, orang-orang manufaktur Lean tidak pernah berhenti mengkhawatirkan ROI untuk mesin mereka. Dan Anda sebaiknya percaya bahwa tidak seorang pun yang sedikit pun khawatir tentang keselamatan atau kemungkinan yang selalu ada bahwa sesuatu yang tidak terduga akan salah. Yang mereka lakukan adalah memperluas alat yang mereka tahu cara menggunakannya dengan baik, dan menciptakan beberapa alat baru, terutama seputar data.

Contoh Lean Construction membantu kita dalam dua cara:

Pertama, ini adalah contoh hebat tentang bagaimana perubahan pola pikir mengubah apa yang dianggap paling berharga oleh para manajer dan pekerja. Dan ketika itu terjadi, proses baru, peluang baru untuk perbaikan pun muncul. Faktanya, Lean menyebar ke seluruh dunia dan mengubah apa yang dianggap berharga oleh semua orang dalam bidang manufaktur.

Bayangkan apa yang terjadi dalam konstruksi sekarang – kita perlahan-lahan mengadopsi pola pikir digital, dan ini membuka cara baru dalam memandang segala hal. Solusi dan teknologi spesifik akan datang dan pergi; yang penting adalah bagaimana Anda, profesional konstruksi, mampu melihat teknologi ini dan membuatnya bekerja untuk Anda. Dalam kasus Lean Manufacturing, lebih dari tujuh dekade teknologi telah mengubah segalanya tentang apa yang terjadi di pabrik, tetapi ide dan metode intinya sama seperti pada tahun 1950-an. Cara Anda berpikir tentang dunia memengaruhi apa yang dapat Anda lakukan; apa yang Anda hargai mendorong apa yang Anda pilih untuk lakukan.

Kedua, Lean menggambarkan sebuah konsep penting – beberapa alat terpenting yang Anda gunakan ada di dalam pikiran Anda sendiri. Dalam kasus Lean, alat-alat tersebut termasuk penggunaan data dan statistik, tetapi juga kebiasaan bertanya kepada orang-orang di lapangan tentang apa yang sedang terjadi, alih-alih bersikap top-down, perintah dan kendali. Alat-alat yang sama tersebut digunakan dalam Konstruksi Lean, seperti yang akan kita bahas nanti. Pola pikir adalah hal yang hebat. Kami menciptakan pola pikir digital yang akan mengantar masuk era konstruksi baru.

1.6 ERA BARU KONSTRUKSI

Terkadang kita bisa melupakan kekuatan konstruksi yang luar biasa. Ya, ini adalah industri yang besar, ya, dan mempekerjakan banyak orang. Namun, begitu juga ritel dan real estat. Konstruksi bukan hanya besar, tetapi juga fundamental, dan penting bagi kita untuk terus meningkatkannya.

Tidak seperti sektor aktivitas manusia lainnya, konstruksi memungkinkan kebutuhan paling dasar kita, dan aspirasi tertinggi kita. Konstruksi menjinakkan alam, menciptakan keamanan dan prediktabilitas yang memungkinkan segala sesuatu menjadi mungkin. Konstruksi juga mendorong bangunan kita ke awan, menciptakan tontonan yang menginspirasi kita untuk percaya bahwa tidak ada yang mustahil.

Masyarakat membutuhkan konstruksi, dan di dunia yang diciptakan kembali oleh pandemi 2020, kita akan membutuhkan hal-hal baru dari konstruksi, dan konstruksi akan membutuhkan hal-hal baru dari teknologi.

Konstruksi akan berubah, mungkin lebih cepat dalam lima tahun ke depan daripada dalam lima puluh tahun terakhir. Tentu saja, konstruksi telah berevolusi dan berubah selama ini, tetapi perubahan tersebut lambat dan tidak merata di seluruh industri. Misalnya, peran arsitek telah berubah dari "master builder" menjadi desainer, yang membuat kontraktor umum menanggung risiko yang mungkin awalnya tidak mereka inginkan. Kesenjangan informasi yang ditinggalkan arsitek telah mengubah cara RFI1 terjadi, cara perubahan order terjadi, dan seterusnya.

Demikian pula, desain-penawaran-bangun tradisional, meskipun masih berlaku untuk banyak proyek, kini berada di bawah tekanan di seluruh dunia, karena pendekatan Desain-Bangun dan Pengiriman Proyek Terpadu dieksplorasi. Namun, sebagian besar perubahan ini tidak sampai pada cara bangunan dibangun. Sebelum sekitar tahun 2010, transformasi digital besar-besaran yang telah melanda setiap industri mulai dari perbankan hingga pertanian hampir tidak menyentuh konstruksi.

Namun, dalam dekade terakhir, banyak hal telah mulai menjadi digital di lapangan. Menurut James Benham dari JBKnowledge: "Semuanya didorong oleh konsumerisasi teknologi. Pekerja yang memiliki iPhone dan aplikasi dan dapat melakukan facetime dengan putri mereka yang tinggal tiga negara bagian jauhnya mengharapkan setidaknya tingkat teknologi itu di lokasi kerja. Ketika mereka tidak dapat menemukannya dari perusahaan teknologi, mereka mulai membuatnya sendiri."

Itulah sebagian dari perubahan pola pikir yang sedang kita bicarakan. Perubahan ekspektasi, perubahan rasa kompetensi digital yang memungkinkan pengawas dan pekerja yang belum tentu ahli teknologi karena pelatihan merasa cukup baik untuk menciptakan solusi mereka sendiri. Mereka merasa demikian karena mereka menggunakan teknologi digital sepanjang hari di luar konstruksi, jadi tampak jelas bahwa mereka harus menggunakannya dalam pekerjaan. Namun, pola pikir konsumerisme itu hanyalah itu, dari ranah konsumen.

Untuk benar-benar membuat perbedaan yang telah kita lihat di industri lain, transformasi digital mengharuskan adanya perubahan pola pikir dari profesional konstruksi yang mencoba-coba iPhone mereka, menjadi profesional konstruksi digital yang bekerja dalam perangkat lunak dan teknologi lain dari bawah ke atas. Kita perlu mengembangkan keterampilan untuk memahami jenis masalah yang ingin kita pecahkan, dan menggunakan alat yang tepat, baik intuitif atau digital atau kombinasi keduanya, untuk memecahkan masalah dan menyelesaikan pekerjaan.

1.7 PERUBAHAN AKAN DATANG

Teknologi akan tetap ada, dan akan semakin menjadi bagian dari proses konstruksi. Seperti yang telah kita bahas sebelumnya, produk tertentu mungkin akan berhasil atau tidak, tetapi transformasi digital telah hadir dan akan menjangkau setiap aspek proses pembangunan.

Penggerak perubahan adalah apa yang terjadi di dunia di luar konstruksi. Buku ini ditulis tepat selama pandemi terbesar dalam 100 tahun, dan meskipun kita tidak akan banyak membahas Covid-19, krisis ini telah menambah bahan bakar pada argumen yang ada bahwa status quo perlu dipikirkan ulang.

Mari kita lihat empat penggerak perubahan eksternal: demografi, perubahan iklim, tekanan pemilik, dan dampak pandemi 2020.

Penggerak Perubahan #1: Demografi

Dalam beberapa tahun mendatang, Generasi Baby Boomer akan terus meninggalkan industri ini karena mereka pensiun, tidak dapat melakukan pekerjaan secara fisik, atau karena alasan lain. Generasi Boomer adalah generasi terbesar dalam sejarah, kecuali anak-anak mereka, generasi milenial – jumlahnya banyak. Generasi baby boomer biasanya dianggap sebagai mereka yang lahir antara tahun 1946 dan 1965, Generasi X adalah mereka yang lahir antara tahun 1966 hingga 1985, dan Generasi Milenial adalah mereka yang lahir antara tahun 1986 dan 2005.

Generasi X secara keseluruhan jumlahnya lebih sedikit beberapa juta jiwa daripada Generasi Milenial atau Generasi Baby Boomer, yang berarti Generasi Milenial akan berada dalam peran pengambilan keputusan jauh lebih awal dalam karier mereka daripada yang seharusnya. Dan yang menjadi masalah tentang Generasi Milenial adalah mereka tidak hanya "melek teknologi" – karena tumbuh dengan internet dan telepon pintar, Generasi Milenial tidak toleran terhadap tempat kerja yang tidak menyukai teknologi. Bagi perusahaan yang ingin tumbuh, atau sekadar menggantikan Generasi Baby Boomer yang sudah tidak lagi menggunakan teknologi, tetap menggunakan teknologi pra-digital akan berdampak serius dalam hal kemampuan untuk menarik dan mempertahankan bakat muda.

Demografi seperti gelombang yang bergerak lambat yang biasanya diabaikan orang, meskipun mereka dapat dengan mudah melihatnya datang. Dan seperti gelombang laut yang besar, demografi tidak dapat dihindari. Gelombang demografi Generasi Milenial yang haus teknologi ini akan berdampak besar pada industri konstruksi, dan tidak hanya pada sisi adopsi teknologi. Sebagai contoh, Generasi Milenial membeli rumah dengan tingkat yang jauh lebih rendah daripada Generasi X atau Generasi Baby Boomer, dan kita sudah melihat fakta itu mengubah bentuk pinggiran kota dan kota, perubahan yang hanya akan semakin cepat saat Generasi Baby Boomer pensiun.

Hal terpenting yang dapat diambil adalah, entah karena mereka menuntutnya, berinovasi, atau berada dalam posisi untuk membuat keputusan untuk membelinya, semakin banyak teknologi akan digunakan di lokasi kerja dan di seluruh rantai nilai konstruksi karena Generasi Milenial.

Namun, demografi memiliki dampak lain di seluruh dunia – perubahan radikal di bagian dunia yang memiliki cukup banyak orang usia kerja, dan bagian yang terlalu sedikit. Dalam 200 tahun terakhir, setiap populasi nasional telah melalui periode kemakmuran, kemudian penurunan perlahan karena penyediaan layanan kesehatan dan sanitasi dasar menyebabkan ketidakseimbangan antara penurunan angka kematian dan angka kelahiran yang masih tinggi, diikuti oleh penurunan angka kelahiran yang hampir universal selama

beberapa dekade ke tingkat yang terlalu rendah bagi penduduk asli untuk menggantikan dirinya sendiri. Pola ini menyebabkan menyusutnya populasi di sebagian besar Eropa, Jepang, dan segera, Tiongkok. Sebaliknya, Afrika mengalami ledakan populasi, dan akan terus demikian selama beberapa tahun. Penuaan penduduk di wilayah utara dan pertumbuhan pesat di wilayah selatan akan berdampak besar pada lokasi pembangunan, cara pembangunan, dan jumlah anak muda yang tersedia di pasar lokal untuk bekerja di bidang konstruksi. Sebagai industri yang menghadapi kekurangan tenaga kerja selama bertahun-tahun, hal ini akan menjadi pendorong lain adopsi teknologi saat kita berupaya untuk melengkapi pekerja manusia dengan berbagai jenis teknologi.

Jauh dari otomatisasi yang mengambil alih pekerjaan, di Eropa, Amerika Utara, dan sebagian besar Asia, otomatisasi akan menyelamatkan pekerjaan dengan memungkinkan proyek-proyek yang tidak mungkin terlaksana tanpanya.

Pendorong Perubahan #2: Perubahan Iklim

Di Amerika Serikat, 40% gas rumah kaca berasal dari lingkungan binaan. Secara global jumlahnya serupa, yaitu 39%. Industri konstruksi diproyeksikan akan menghasilkan 2,2 miliar ton limbah pada tahun 2025. Itu adalah jumlah yang sangat besar, dan tidak dapat dipertahankan selamanya. Dalam beberapa tahun terakhir, misalnya, banyak negara berkembang telah berhenti menerima limbah padat AS dan Eropa, yang memaksa mereka untuk membuat pilihan sulit tentang di mana akan membuangnya. Tumpukan sampah Pasifik yang besar, kumpulan plastik kecil dan besar yang mengapung seluas sekitar 1,6 juta kilometer persegi, adalah bukti bahwa kekhawatiran tentang tempat kita membuang sampah telah berubah dari sekadar peringatan abstrak para pencinta lingkungan menjadi kenyataan konkret dan praktis yang mengancam pasokan makanan kita, serta kesehatan kita yang sebenarnya.

Baik dari pemerintah, kelompok yang peduli, atau investor, tekanan untuk mengurangi limbah padat dari konstruksi akan terus meningkat di tahun-tahun mendatang.

Namun, perubahan iklim bukan hanya pendorong negatif. Meskipun konstruksi berkontribusi terhadap perubahan iklim, konstruksi juga merupakan garis pertahanan pertama kita terhadap banyak bahaya yang akan ditimbulkan oleh perubahan iklim. Karena daerah dataran rendah di Italia, Belanda, dan di AS, termasuk sebagian besar Florida selatan dan negara bagian teluk Alabama, Mississippi, dan terutama Louisiana menghadapi prospek kenaikan permukaan laut, konstruksi akan membangun tembok, tanggul, dan drainase yang akan menyelamatkan masyarakat kita dari gelombang badai yang semakin mengancam mereka. Mudah untuk melihat proyeksi seluruh wilayah negara yang akan terendam air dalam 50–100 tahun ke depan dan menganggap remeh proyeksi ini sebagai sesuatu yang tidak pasti. Namun, kita tidak dapat mengabaikan fakta bahwa ancaman tidak harus terjadi jika daratan berada di bawah air – badai seperti Katrina dan Sandy menunjukkan seberapa besar kerusakan yang dapat ditimbulkan oleh badai selama dua hari terhadap lautan di tempat yang sama seperti sekarang.

Pendorong Perubahan #3: Tuntutan dari Pemilik

Pemilik dan pengembang semakin banyak menuntut dalam beberapa tahun terakhir, baik untuk peningkatan keselamatan, atau baru-baru ini, pengembangan tenaga kerja.

Organisasi seperti Construction Users Round Table (CURT) sering kali memimpin dalam mendorong standar baru, dan mekanisme sertifikasi yang menyebabkan pemilik memerlukan upaya yang lebih tinggi untuk mencapai sertifikasi ini. Dan karena semakin banyak perusahaan teknologi menjadi pemilik aset lingkungan yang dibangun, mereka menuntut penggunaan teknologi manajemen proyek dan lapangan yang setara dengan bidang lain dalam bisnis mereka.

Pendorong Perubahan #4: Respons Pandemi 2020

Dalam beberapa minggu setelah pembatasan wilayah akibat pandemi Covid-19, perusahaan konstruksi dan penyedia teknologinya menyesuaikan diri dengan kebutuhan menjaga jarak sosial dengan menyelenggarakan lebih banyak rapat dari jarak jauh, dan mulai menerapkan teknologi penginderaan di lokasi untuk memantau jarak sosial.

Perusahaan seperti StructionSite, yang menyediakan manajemen foto 360 derajat di lokasi kerja, menyediakan layanan mereka secara gratis atau dengan biaya yang lebih rendah untuk klien konstruksi yang menyesuaikan diri dengan pembatasan sosial, dan yang tiba-tiba tidak lagi bersemangat untuk mengirim staf yang tidak penting ke mana pun. Demikian pula, seperti orang lain, tim konstruksi menjadi jauh lebih bersedia untuk mengadakan rapat melalui perangkat lunak telekonferensi digital. Setelah berbulan-bulan, banyak yang telah mengubah alur kerja dan proses secara permanen menjadi lebih digital dan jarak jauh.

Panjangnya persyaratan pembatasan sosial selama karantina wilayah dan pembukaan kembali secara bertahap berarti bahwa banyak proses akan berubah selamanya, dan tahun-tahun mendatang akan melihat dampak dari perubahan itu ketika fitur-fitur baru pada produk yang sudah ada, serta produk yang sama sekali baru, diperkenalkan untuk mengatasi perubahan ini, dan dalam beberapa kasus memungkinkan alur kerja baru. Sebagai contoh, kami berharap bahwa realitas virtual dan tertambah akan menemukan penerimaan dan adopsi yang lebih cepat di dunia di mana kebersamaan secara fisik jauh lebih tidak dihargai daripada sebelumnya. Di berbagai industri, kami telah melihat banyak perusahaan melaporkan bahwa pandemi Covid-19 telah menyebabkan adopsi teknologi selama satu dekade terjadi hanya dalam beberapa bulan. Hal ini telah menyebabkan asumsi yang dijunjung tinggi, seperti kebutuhan untuk berada di tempat yang sama secara fisik, ditantang dan terbukti salah, sehingga asumsi lain terbuka untuk dievaluasi ulang. Kami belum dapat mengetahui dampak akhir dari perubahan yang begitu besar, tetapi Covid-19 telah menyebabkan perubahan besar dalam penggunaan teknologi konstruksi.

1.8 PASOKAN TEKNOLOGI

Pada saat yang sama ketika faktor-faktor ini menekan para pelaku industri konstruksi untuk mengadopsi teknologi baru, beberapa tahun terakhir telah melihat banjir produk teknologi baru, terutama perangkat lunak. Segala sesuatu mulai dari cara baru untuk membuat, berbagi, dan menggunakan Building Information Models (BIM), hingga serangkaian solusi manajemen proyek, hingga drone, perangkat lunak yang dioperasikan dengan suara, dan banyak lagi telah tiba-tiba tersedia. Sepanjang perjalanan, komunitas modal ventura dan investor lain telah "menemukan" teknologi konstruksi, yang mengarah ke semakin banyak

pilihan. Meskipun banjir ini telah menyebabkan beberapa perusahaan merasa bahwa ini terlalu cepat, perusahaan konstruksi telah menciptakan tim yang berfokus pada keberhasilan menavigasi semua opsi, mengadopsi teknologi baru ini, dan membuatnya bekerja untuk tim mereka di lapangan dan di kantor.

Dan dengan semua orang ini yang berusaha keras untuk menciptakan solusi baru, beberapa hal yang benar-benar berguna akan hadir di pasaran. Tidak berlebihan untuk mengatakan bahwa produk seperti Procore telah mengubah cara kerja dikelola dan ditangani di lokasi kerja. Alat perekaman alur kerja dan lokasi kerja seperti Holobuilder mengubah tingkat kesadaran perusahaan terhadap pekerjaan mereka di seluruh ruang dan waktu dengan cara yang benar-benar bermakna.

Beberapa tahun terakhir telah terjadi evolusi dari solusi titik yang kikuk dan terputus-putus menjadi pendekatan yang lebih berjejaring di mana banyak solusi perangkat lunak benar-benar bekerja bersama. Namun, menghubungkan setiap produk perangkat lunak secara benar-benar bersama-sama tetap menjadi masalah yang sulit dalam konstruksi, dengan banyak perusahaan yang meretas solusi bersama-sama, atau mengalami masalah entri data ganda dan tiga kali lipat yang menyakitkan yang telah mengganggu industri selama bertahun-tahun. Semua solusi ini hadir di industri konstruksi karena solusi tersebut, atau solusi yang serupa, sudah digunakan di industri lain – konstruksi, karena alasan yang sah, telah menjadi pengadopsi teknologi yang lambat dalam sebagian besar sejarahnya. Manfaat menjadi yang kedua – atau mari kita hadapi, lebih seperti seperlima – bagi kelompok teknologi di balik industri lain adalah bahwa perusahaan konstruksi dan pekerja dapat belajar dari pemikiran selama puluhan tahun tentang transformasi digital dan memilih ide-ide terbaik. Buku ini akan membantu Anda melakukannya.

Penggunaan teknologi adalah sebuah teknologi

Setiap kali Anda menggunakan perkakas listrik, atau perkakas non-listrik, Anda menerapkan pengetahuan, dan itu adalah sebuah teknologi. Proses, pengetahuan, ini semua adalah sejenis teknologi, semuanya adalah sarana untuk menyelesaikan sesuatu demi suatu tujuan.

Setiap perkakas baru mengharuskan Anda mempelajari beberapa hal. Perkakas tangan jenis baru memerlukan latihan, dan percobaan di dunia nyata untuk memahami cara menggunakannya. Ini, sekali lagi, adalah sejenis teknologi. Namun, untuk perkakas non-digital semacam ini, tidak ada gunanya menyebutnya "teknologi," melainkan hanya disebut "pengetahuan."

Alasan saya menyebut definisi teknologi ini, sebagai cara melakukan sesuatu, bukan hanya apa yang harus digunakan, adalah karena tidak seperti perkakas non-digital, perangkat lunak dan solusi digital bekerja dengan cara yang sangat dapat diprediksi dan sempit yang harus Anda pelajari dengan cara yang sangat berbeda dari cara Anda mempelajari perkakas non-digital.

Tidak seorang pun akan menyarankan Anda untuk belajar menggunakan bor tanpa pernah memegangnya. Namun, Anda dapat mempelajari banyak hal tentang Procore tanpa harus membuka platform tersebut, melalui eLearning mereka. Anda dapat mempelajari

semua hal yang perlu Anda ketahui tentang Revit dari tutorial dan video, dan hal yang sama juga berlaku untuk hampir semua produk berbasis perangkat lunak.

Menurut saya, teknologi untuk pembelajaran merupakan salah satu perubahan terbesar yang terjadi di bidang konstruksi. Pembelajaran ini semudah menonton video YouTube, dan dapat dilakukan dari ponsel, desktop, atau tablet di mana saja.

Di masa lalu, tenaga kerja didorong untuk "mengadopsi" berbagai hal seolah-olah itu hanya satu perubahan besar, maka Anda akan kembali ke keadaan normal yang stabil. Dan mungkin itu benar pada tahun 1970-an, tetapi itu tidak benar lagi sekarang.

Perangkat lunak tidak diperbarui setiap beberapa tahun dan dikirimkan kepada Anda seperti Windows pada tahun 1990-an. Itu adalah situs web yang dihosting di tempat lain, di "awan". Itu terus diperbarui, didesain ulang, diganti. Prolog tahun 2010-an adalah Sudut Pandang masa kini, dan Procore masa depan tidak akan terlihat seperti Procore masa kini, karena akan terus berkembang.

Hal ini menimbulkan risiko bagi orang biasa yang tidak benar-benar ingin fokus pada teknologi, tetapi sebenarnya suka melakukan pekerjaan mereka yang sebenarnya, baik itu membuat rangka dinding, menyiapkan sistem mekanis, atau memasang sistem kelistrikan. Solusinya adalah memahami dasar-dasar cara kerja perangkat lunak, AI, BIM, Prefab, dan teknologi lainnya.

BAB 2
PERANGKAT LUNAK

“Perangkat Lunak Menguasai Dunia.”
– Marc Andreeson,

Di seluruh perekonomian, industri telah melakukan digitalisasi. Perusahaan telah mengubah proses analog berbasis kertas menjadi pendekatan yang berpusat pada perangkat lunak, hingga titik di mana inovasi dalam perangkat lunak lebih penting bagi keberhasilan dalam industri ini daripada hal lainnya. Dari ritel hingga manufaktur hingga perjalanan, perangkat lunak telah menjadi arena untukpersaingan dan peningkatan efisiensi.

Tren ini mencapai puncak visibilitas dengan digitalisasi transportasi dengan Uber dan Lyft, dan kamar hotel dengan AirBnB, bersama dengan sejumlah besar pekerjaan "ekonomi pertunjukan" lainnya.

Tren digitalisasi yang sama telah terjadi dalam konstruksi selama bertahun-tahun, meskipun perjalanannya berliku-liku. Banyak orang di industri ini akan memberi tahu Anda bahwa perangkat lunak pertama kali memasuki industri konstruksi melalui perangkat lunak akuntansi, dan sering kali tidak mudah digunakan. Beberapa tahun terakhir telah menyaksikan munculnya solusi yang lebih modern seperti Procore, BIM360, dan banyak solusi yang lebih terfokus seperti Rhumbix, Raken, dan eSub. Yang kita tahu pasti adalah bahwa perangkat lunak memang memberikan nilai tambah; bahkan dapat mengubah industri, meskipun memerlukan beberapa penyesuaian.

Pada saat yang sama, banyak perangkat lunak yang sulit digunakan, sulit diintegrasikan, dan tidak selalu melakukan apa yang kita butuhkan. Mengetahui bahwa manfaat yang dijanjikan tidak benar-benar terwujud, atau bahwa perangkat lunak sama sekali tidak berfungsi, membutuhkan usaha dan sumber daya. Memahami cara menilai dan menggunakan perangkat lunak sangatlah penting.

Ketika kita berbicara tentang "teknologi dalam konstruksi," hal pertama yang akan dipikirkan kebanyakan orang adalah perangkat lunak. Seperti yang ditunjukkan oleh kutipan di atas, perangkat lunak kini ada di mana-mana, mulai dari alat di tangan Anda hingga jadwal di laptop Anda hingga desain 3D yang mengubah cara kita memahami dan membangun dunia kita. Bab ini akan memperkenalkan perangkat lunak dari dasar, memberi Anda alat untuk memahami apa yang dapat dan tidak dapat dilakukan oleh perangkat lunak, dan cara mengajukan pertanyaan yang tepat untuk memastikan Anda mampu menguasai perangkat baru yang terus berkembang. Kita mulai, seperti yang kita lakukan dengan teknologi, dengan definisi dasar, lalu membahas konsep-konsep utama yang membuat perangkat lunak modern berfungsi dan yang perlu Anda ketahui untuk memahami cara perangkat lunak modern menjalankan tugasnya.

2.1 PRINSIP DASAR PERANGKAT LUNAK

Sama seperti beberapa kata lain yang telah kita bahas sebelumnya, perangkat lunak sering digunakan, tanpa ada yang meluangkan waktu untuk benar-benar mendefinisikannya. Mari kita bahas di sini – saya suka definisi Wikipedia:

> *Perangkat lunak komputer, atau hanya perangkat lunak, adalah kumpulan data atau instruksi komputer yang memberi tahu komputer cara bekerja. Ini berbeda dengan perangkat keras fisik, yang menjadi dasar sistem dibangun dan benar-benar melakukan pekerjaan. (Wikipedia)*

Tiga hal yang menonjol dari definisi itu: data, instruksi, dan perangkat keras. Anggap data sebagai material, instruksi sebagai proses, dan perangkat keras sebagai … perangkat keras, dan Anda memiliki mesin seperti halnya hal lainnya. Data adalah apa yang digunakan perangkat lunak – instruksi ditulis dalam kode yang mendefinisikan apa yang dapat dilakukan perangkat lunak, seperti roda gigi dalam mesin fisik – dan perangkat keras adalah mesin, seperti iPhone atau laptop. Baik perangkat lunak berjalan di iPhone, laptop, atau di server di suatu tempat, fungsinya adalah menerima informasi dari dunia, menerjemahkan informasi tersebut dari cara terjadinya di dunia ke cara komputer harus menanganinya, meneruskan informasi yang mudah dipahami komputer ke perangkat keras, tempat instruksi yang diprogram oleh sekelompok orang dijalankan, lalu meneruskan hasilnya ke dunia, biasanya muncul di layar di suatu tempat.

Bisnis pembuatan perangkat lunak memiliki banyak kesamaan dengan pembuatan bangunan. Ada pemilik yang memutuskan bahwa mereka perlu membangun perangkat lunak, dan mereka menyewa seorang arsitek untuk merancang perangkat lunak, menentukan beberapa tetapi tidak semua detailnya. Insinyur kemudian datang dan merancang sebagian besar perangkat lunak lainnya, dengan keahlian khusus mengisi hal-hal tertentu seperti pengalaman pengguna, keberadaan web, dan konektivitas API.

Perangkat lunak selalu tidak kompeten. Perangkat lunak melakukan beberapa hal dengan sangat baik, tetapi tidak melakukan hal-hal lain yang biasanya kita anggap seharusnya dilakukan. Setiap orang yang pernah saya tanyai memiliki cerita tentang perangkat lunak yang entah bagaimana mengecewakan mereka. Perangkat lunak modern, karena biasanya berubah hampir terus-menerus, sering kali lebih kompeten daripada perangkat lunak masa lalu, tetapi menurut definisinya masih sangat terbatas.

Sebelumnya kita membahas tentang pentingnya pola pikir, dan cara kita berpikir tentang sesuatu memengaruhi apa yang dapat kita lakukan dengannya. Memikirkan perangkat lunak sebagai mesin, yang dibuat untuk melakukan hal-hal tertentu, tetapi hanya hal-hal tersebut, membantu kita memahami cara menghindari frustrasi karena menganggap perangkat lunak lebih dari itu.

2.2 PERANGKAT LUNAK ADALAH MESIN

Perangkat lunak seperti mesin lain yang Anda gunakan untuk melakukan pekerjaan. Perangkat lunak adalah seperangkat alat yang memungkinkan Anda melakukan lebih banyak

hal daripada yang dapat Anda lakukan tanpanya. Perangkat lunak seharusnya membuat pekerjaan Anda lebih cepat, memungkinkan Anda mengotomatiskan beberapa hal, atau setidaknya membuat Anda lebih aman.

Dalam beberapa tahun mendatang, perangkat lunak akan terus menjadi semakin pintar – suatu hal yang akan kita bahas lebih lanjut dalam bab kecerdasan buatan. Namun, untuk saat ini, mari kita bahas tentang batasan dan kemungkinan perangkat lunak.

Seperti yang kita ketahui sebelumnya, perangkat lunak adalah seperangkat instruksi. Seseorang harus memikirkan apa yang harus dilakukannya, menulis instruksi tersebut, dan menerbitkannya. Instruksi ini persis seperti roda gigi dalam mesin – instruksi ini memungkinkan perangkat lunak melakukan serangkaian hal sebagai respons terhadap tindakan pengguna. Namun, seperti halnya roda gigi dalam mesin, perangkat lunak tidak dapat melakukan satu hal pun yang tidak dirancang untuk dilakukannya.

Hal ini dapat membuat kita tersandung, karena hampir tidak ada hal lain di dunia ini yang memiliki keterbatasan seketat itu. Mesin fisik melibatkan manusia sebagai bagian dari operasinya yang menyediakan fleksibilitas penting yang tidak dimiliki perangkat lunak, dan tentu saja manusia sebenarnya pada dasarnya fleksibel dan mampu beradaptasi secara otomatis terhadap masukan yang tidak jelas, situasi yang tidak jelas, dan skenario baru. Orang mungkin hanya memenuhi syarat untuk melakukan pekerjaan yang telah mereka latih, tetapi tanpa banyak kesulitan mereka dapat membantu di area lain, bahkan area yang tidak mereka pahami dengan baik. Perangkat lunak tidak melakukan semua ini – kecuali ada manusia yang terlibat di suatu tempat, perangkat lunak tidak dapat menyesuaikan diri dengan perintah yang tidak jelas atau situasi baru sama sekali.

Hal ini penting karena ketika kita membuat perangkat lunak, hal pertama yang selalu kita lakukan adalah membuat sketsa persyaratan yang sangat spesifik. Dan Anda sebagai pembeli atau pengguna perangkat lunak perlu melakukan hal yang sama.

Cara Kami Membuat Perangkat Lunak

Faktanya, cara perangkat lunak dibuat mencerminkan kebutuhan untuk memenuhi persyaratan dengan benar. Langkah pertama tentu saja adalah memutuskan sesuatu yang dibutuhkan, jadi Anda duduk bersama orang-orang yang akan menggunakan sesuatu itu dan memutuskan secara kasar apa yang mereka butuhkan dan inginkan, dan fungsi apa yang akan dibutuhkan untuk melakukannya.

Di situlah Anda akan memutuskan jenis perangkat lunak apa yang akan dibuat, apakah itu aplikasi seluler atau aplikasi berbasis cloud, apakah itu memerlukan basis data besar, dan seterusnya. Pada titik ini, proses yang baik adalah membuat prototipe. Prototipe penting karena memberi Anda sesuatu yang nyata untuk dicoba, dan pikiran selalu bekerja lebih baik dalam memahami sesuatu ketika ada sesuatu yang nyata. Perlu diingat, prototipe tidak melakukan semua fungsi yang pada akhirnya akan Anda buat, tetapi terlihat dan berfungsi seperti perangkat lunak itu sendiri. Lihat www.proto.io, www.balsamiq.com, atau www.invisionapp.com jika Anda ingin melihat beberapa contoh alat pembuatan prototipe yang hebat. Selanjutnya, prototipe harus diuji dengan calon pengguna, sebelum apa pun dibangun. Ini penting dan banyak perusahaan rintisan yang melewatkan langkah ini karena

mereka terburu-buru atau tidak tahu cara menemukan penguji. Namun, jika Anda tidak menguji dengan pengguna terlebih dahulu, Anda tidak dapat memastikan produk tersebut dibutuhkan, bahwa orang sungguhan akan mengerti kegunaannya, atau bahwa Anda memiliki arahan desain yang benar.

Setelah kami mencoba prototipe baik secara internal maupun dengan pengguna, kami kembali ke papan gambar dan membuat perubahan berdasarkan apa yang telah kami pelajari. Ini biasanya berkaitan dengan tombol mana yang harus diklik, urutan interaksi mana yang lebih alami, dan fitur apa yang mungkin bagus untuk ditambahkan.

Pengembangan Agile

Apa yang terjadi selanjutnya adalah apa yang kami sebut metodologi Agile. Ini sebenarnya adalah cara sebagian besar perangkat lunak modern dibangun, dan penting bagi Anda sebagai teknolog konstruksi untuk memahami cara kerjanya dan mengapa ia bekerja seperti itu.

Di masa lalu, perangkat lunak akan direncanakan seperti bangunan. Setiap fitur dan fungsi akan dipetakan, dijadwalkan pada hari itu, dan dikemas dalam apa yang mereka sebut "rencana air terjun," karena setiap langkah dalam proses tersebut mengalir ke langkah berikutnya secara teratur.

Dalam praktiknya, ini berarti bahwa tim yang melakukan perencanaan akan menyusun rencana mereka, kemudian mengirim para insinyur pergi selama berbulan-bulan. Para insinyur akan kembali dengan perangkat lunak baru mereka yang mengilap, biasanya lebih lambat dari yang dimaksudkan, tetapi yang terpenting, perangkat lunak itu tidak akan kembali seperti yang diharapkan oleh tim perencana.

Ternyata, seperti dalam konstruksi, ada banyak sekali keputusan kecil yang dibuat di sepanjang jalan ketika Anda membangun perangkat lunak, dan jika ini hanya dibuat oleh tim internal, pada akhirnya, sedikit demi sedikit, Anda akan menjauh dari niat pertama. Tidak seperti konstruksi, tidak ada kontrak yang menetapkan rencana yang menjadi kriteria pengiriman, tidak ada RFI saat keputusan ini dibuat – tim perangkat lunak hanya membuat keputusan terbaik yang bisa diambil dan melanjutkan.

Pendekatan ini menyebabkan penundaan dan biaya yang membengkak, dan yang terpenting, memastikan bahwa keputusan penting tentang produk akhir tidak dibuat dengan mempertimbangkan pemilik proyek atau pengguna akhir.

Jadi, pada bulan Februari 2001, sekelompok insinyur papan atas menerbitkan cara berpikir baru tentang pengembangan perangkat lunak, dalam Agile Manifesto. Di agilemanifesto.org Anda dapat melihat 12 prinsip manifesto tersebut. Ini memfokuskan pikiran pada pembuatan perangkat lunak yang berfungsi lebih awal dan sering, dan mendapatkan umpan balik bisnis di sepanjang jalan.

Pada tahun-tahun sejak publikasi itu, banyak pemikiran dan proses telah berkembang di sekitar proses agile, dan telah menjadi beberapa bentuk. Saya cenderung menjalankan versi Agile yang lebih ringan karena tim yang saya pimpin tidak cukup seukuran Facebook atau Microsoft. Prosesnya dimulai selama fase pembuatan prototipe, atau tepat saat Anda telah

mengumpulkan masukan dari pengguna. Ada rapat awal, tempat jadwal dan peran ditetapkan. Ini membantu dan tidak jauh berbeda dari proyek normal.

Kemudian kami sepakati jadwal pertemuan klien, biasanya setiap dua minggu. Periode dua minggu ini disebut sprint, dan pada awalnya disepakati dengan pengembang perangkat lunak bahwa akan ada serangkaian persyaratan yang akan mereka kerjakan, dan spesifikasi ini tidak akan berubah selama sprint dua minggu ini. Ini memungkinkan tim untuk fokus, dan mempercepat banyak hal.

Sprint ini harus menghasilkan perangkat lunak yang berfungsi, dan klien perlu menggunakan perangkat lunak tersebut dan memberikan umpan balik yang aktif dan konkret. Ini melakukan beberapa hal: pertama, ini menghentikan pengembang dari tersesat dalam blok kode yang besar. Kedua, ini memastikan bahwa klien mampu mengadaptasi persyaratan mereka ke perangkat lunak nyata, bukan ide abstrak atau prototipe yang mereka mulai. Dan terakhir, ini membuat tim bekerja sama menuju tujuan yang dipahami bersama.

Di antara pertemuan sprint ini ada pertemuan harian tim, yang disebut "standup." Inti dari standup adalah untuk menghindari pertemuan yang panjang dan berlarut-larut yang sering terjadi – Anda benar-benar membuat semua orang berdiri sehingga secara alami menjadi pertemuan yang lebih singkat. Setiap orang membicarakan apa yang sedang mereka kerjakan, masalah apa pun yang muncul, pertanyaan atau kebutuhan. Kemudian semua orang mulai bekerja.

Maka, inti dari Agile adalah menjaga hubungan erat antara apa yang dibutuhkan klien atau pengguna, dan apa yang dibangun oleh tim.

2.3 ANATOMI PERANGKAT LUNAK

Kita menganggap perangkat lunak berlapis-lapis, jadi di sini kita akan membahas cara kerjanya, dan apa saja bagian-bagian utamanya, mulai dari layar yang Anda sentuh dan gunakan, hingga perangkat keras yang menggerakkan semuanya. Bagian ini sengaja disusun agar mudah dirujuk nanti, dan akan dirujuk dalam indeks untuk tujuan tersebut.

Pengguna

Pengembangan perangkat lunak mengacu pada siapa saja yang benar-benar menggunakan perangkat lunak sebagai "pengguna." Ini bisa jadi pengawas di lapangan yang melihat jadwal di iPad, subkontraktor yang membuat laporan harian di ponsel mereka, penaksir yang bekerja dengan spreadsheet di kantor, arsitek yang mendesain keseluruhan bangunan, penentu spesifikasi yang bekerja di Revit untuk merinci kamar mandi, dan banyak lagi.

Apa yang akan dilakukan pengguna sangat sulit untuk dijelaskan, karena kebanyakan dari kita tidak berpikir seperti cara kerja perangkat lunak. Kita terbiasa berurusan dengan manusia lain, dengan pikiran yang dapat mencari tahu apa yang kita maksud meskipun kita tidak begitu spesifik, jelas, atau lengkap dalam apa yang kita katakan.

Sebaliknya, perangkat lunak sangat buruk dalam menebak apa yang Anda maksud. Ingat, itu hanyalah kumpulan instruksi, terlepas dari seberapa profesional desainnya dan tampilannya – pada akhirnya itu hanyalah sebuah "mesin."

Sebagai pengguna individu, terkadang sulit untuk melihat seberapa terbiasanya kita dengan akal sehat manusia lain. Orang-orang, bahkan orang-orang yang mungkin kita pikir tidak memperhatikan, sangat hebat dalam menafsirkan apa yang dimaksud orang lain, dan dapat menebus banyak ambiguitas dengan memahami apa yang sedang terjadi, apa konteksnya.

Setiap pengguna berada dalam suatu konteks. Mereka berada di kota tertentu, pada pekerjaan tertentu, melakukan tugas tertentu, dengan serangkaian kebutuhan dan fakta tertentu yang harus dihadapi. Setiap manusia pasti sudah tahu, atau dapat menebak, sebagian besar poin kontekstual ini. Jadi, kita sebagai manusia menganggap remeh bahwa perangkat lunak akan mampu melakukan ini, karena setiap interaksi lain dalam hidup kita adalah dengan seseorang.

Perangkat lunak tidak tahu tentang konteks. Perangkat lunak hanya tahu apa yang dirancang untuk diterima dari Anda, yang biasanya berupa beberapa klik tombol dan beberapa kata yang diketik.

Analogi yang bagus adalah layanan pelanggan di toko perangkat keras. Petugas layanan pelanggan tidak tahu apa pun tentang Anda sebelum Anda datang ke sana, yang harus mereka lakukan hanyalah apa yang Anda berikan kepada mereka. Terkadang mereka memahami masalah Anda dengan cepat, menyelesaikannya, dan Anda pun pergi. Namun, bagaimana jika Anda kehilangan struk dan mereka memiliki kebijakan pengembalian hanya untuk struk? Petugas layanan pelanggan itu hanya dapat mendengar satu hal: "Saya ingin mengembalikan ini, ini struknya." Tidak ada jumlah cerita yang dapat mengubahnya, perwakilan layanan pelanggan tidak memiliki minat atau kemampuan untuk melakukan apa pun dengan konteks Anda. Anda memiliki apa yang mereka butuhkan untuk melakukan pekerjaan mereka, atau tidak.

Pikirkan analogi itu saat perangkat lunak yang Anda gunakan macet – bukan salah Anda jika perangkat lunak hanya dapat menangani rentang input yang sempit, tetapi tetap saja itu kenyataan. Seni desain pengalaman pengguna adalah mengantisipasi kebutuhan dan menciptakan perangkat lunak yang dapat menerima hampir semua cara yang akan dicoba sebagian besar pengguna untuk berinteraksi dengan perangkat lunak, tetapi mereka akan selalu melewatkan sesuatu. Kami menyebutnya "kasus tepi", saat pengguna melakukan sesuatu yang tidak dapat ditangani perangkat lunak, yang tidak dilakukan kebanyakan orang. Bagian tersulit dari kasus tepi adalah semakin rumit perangkat lunak, semakin banyak cara berinteraksi dengan perangkat lunak yang tidak terduga dan tidak lazim ini. Akibatnya, perangkat lunak harus menjadi semakin rumit untuk menangani kasus tepi ini, dan seterusnya.

Sekitar dua tahun lalu, saya mengembangkan produk yang menggunakan suara sebagai input untuk laporan lokasi konstruksi. Saya telah membuat prototipe yang meminta komputer untuk menghubungi pengguna dan mengajukan serangkaian pertanyaan tertulis. Untuk mempercepat dan mempermudah, banyak dari pertanyaan ini hanya memerlukan jawaban "ya" atau "tidak". Pertanyaan pertama yang akan diajukan komputer adalah "Apakah pekerjaan tertunda hari ini?" Saya telah memprogram sistem untuk menerima jawaban "Ya" atau "Tidak". Kedengarannya jelas, bukan? Saya meminta seorang teman baik, Damon

Hernandez, untuk menguji sistem di awal pengembangan. Damon telah membangun perangkat lunak selama bertahun-tahun dan ingin membantu saya meningkatkan perangkat lunak tersebut. Dan Damon memiliki selera humor. Jadi ketika sistem menelepon, Damon menjawab sebagai berikut: "Halo, apakah Anda ingin membuat laporan harian hari ini?" tanya AI yang bersemangat. "Ya," kata Damon yang sabar. "Bagus. Apakah ada penundaan pekerjaan hari ini?" tanya AI yang masih bersemangat. "Sedikit," kata Damon. Keheningan ...

Seperti itu, Damon menunjukkan banyaknya cara orang sungguhan berinteraksi dengan perangkat lunak, karena kita terbiasa berbicara dengan orang lain, dan orang lain jauh lebih fleksibel dan mampu menangani kasus-kasus khusus daripada perangkat lunak.

Semua perangkat lunak tidak memiliki beberapa kasus khusus. Jadi, jika Anda mencoba menyelesaikan sesuatu dengan perangkat lunak baru, sering kali itu berarti Anda hanya memberinya masukan yang tidak sesuai dengannya. Jadi, Anda dapat mencobanya dengan cara yang berbeda, atau mencari di Google cara melakukan apa pun yang ingin Anda lakukan. Saya biasanya menyarankan yang terakhir – sebenarnya tema dalam buku ini adalah bahwa apa pun masalah yang Anda alami dengan sesuatu yang teknis, orang lain juga pernah mengalami masalah itu.

Pengguna tidak "sendirian." Ada banyak kelompok di mana-mana yang mencoba mempelajari, menggunakan, dan mengoptimalkan perangkat lunak. Anda dapat mencarinya di Google, atau Anda dapat mengunjungi situs web perusahaan perangkat lunak atau perangkat keras dan mencari bagian Tanya Jawab Umum (FAQ), dan forum pengguna yang sering mereka selenggarakan. Sebagai pengembang perangkat lunak, saya dapat memberi tahu Anda bahwa kreator perangkat lunak belajar sejak dini bahwa tidak ada pertanyaan yang bodoh; faktanya, pertanyaan bodoh adalah yang paling banyak ditanyakan. Mencari di Google atau pergi ke forum mungkin merupakan cara baru untuk meminta saran, tetapi sebenarnya tidak ada bedanya dengan bertanya kepada teman – dalam hal ini secara anonim jika Anda mau, dan biasanya cukup cepat.

2.4 ANTARMUKA PENGGUNA (UI)

Cara apa pun yang dapat digunakan pengguna untuk memberikan informasi atau masukan ke komputer, dan cara apa pun yang dapat digunakan untuk memperoleh informasi atau keluaran, disebut "antarmuka." Anda terkadang mendengar ini disebut sebagai "UI," untuk "antarmuka pengguna."

Ada berbagai jenis antarmuka yang luar biasa. Kembali pada tahun 1960-an dan 1970-an, antarmuka adalah, percaya atau tidak, kartu berlubang. Pengguna akan memasukkan kartu berlubang ke komputer, komputer akan membacanya, melakukan apa pun yang diperlukan berdasarkan informasi dalam kartu tersebut dan instruksi perangkat lunak komputer itu sendiri, lalu komputer akan memasukkan keluarannya ke kartu yang sama atau kartu yang lain. Kedengarannya memang aneh, tetapi ini membantu menunjukkan sesuatu – antarmuka pengguna digerakkan oleh teknologi yang mendasarinya. Antarmuka pengguna hanya akan sebaik perangkat lunak yang terhubung dengan perangkat tersebut, jadi perangkat lunak dan perangkat keras yang hebat biasanya memiliki antarmuka yang hebat,

dan perangkat lunak/perangkat keras yang terbatas biasanya memiliki antarmuka yang terbatas.

Dalam kasus kartu berlubang, komputer saat itu pada dasarnya tidak memiliki memori, dan hanya dapat melakukan satu operasi dalam satu waktu, jadi kartu berlubang memiliki data dan instruksinya di tempat yang sama. Ada alasan mengapa orang awam tidak banyak mendengar tentang komputer hingga tahun 1980-an.

Saat itulah komputer menjadi cukup kuat untuk menggunakan layar, dan memiliki memorinya sendiri, sekecil apa pun. Seiring berjalannya tahun 1990-an, komputer kita menjadi semakin baik, dan kita beralih dari perintah pengetikan ke antarmuka pengguna grafis (GUI) yang kita anggap biasa sekarang. Namun ketika Apple memperkenalkan Macintosh dengan layar berbasis gambarnya, diikuti oleh Microsoft yang memperkenalkan sistem operasi Windows mereka, dunia berubah. Saat itulah kita mulai dapat menggunakan gambar untuk menggambarkan berbagai hal, seperti tempat sampah atau berkas tertentu, dan kita dapat menarik dan melepasnya, di antara hal-hal lainnya. Hal ini membuat komputer berguna bagi orang-orang biasa.

Kembali pada tahun 1990-an, kebanyakan orang hanya menggunakan beberapa aplikasi, seperti Microsoft Word, Excel, dan beberapa PowerPoint, sehingga kebutuhan akan antarmuka pengguna yang benar-benar intuitif tidak setinggi yang terjadi kemudian.

Kemudian pada tahun 1993, seorang lulusan Universitas Illinois baru-baru ini bernama Marc Andreeson mengambil ide "bahasa markah hiperteks" yang telah ditemukan oleh seorang ilmuwan komputer Inggris bernama Tim Berners-Lee, dan menciptakan peramban Mosaic.

Seiring dengan munculnya peramban web Mosaic, dan AOL serta CompuServe yang menghubungkan banyak orang secara daring, jaringan internet mengubah cara kita menggunakan perangkat lunak dan aplikasi selamanya. Tiba-tiba, kita memiliki banyak hal baru untuk dilakukan, seperti menulis dan mengirim email serta menjelajahi halaman web. Seiring berjalannya waktu, semakin banyak tugas luring yang dilakukan secara daring, mulai dari berbelanja hingga merencanakan perjalanan, membayar pajak, dan sebagainya.

Hal ini mendorong peningkatan besar dalam pentingnya antarmuka pengguna, dan karena ada begitu banyak aplikasi baru, kita harus mulai memikirkan tentang cara pengguna mengeklik, mengetik, dan menonton video dalam perangkat lunak. Apa yang paling sering mereka lihat? Apa yang mereka abaikan? Instruksi penting apa yang mereka lewatkan?

Sebelumnya, kita telah mempelajari bagaimana orang terbiasa berinteraksi dengan orang lain, sehingga mereka tidak secara alami menanyakan aturan apa yang diperlukan untuk membuat diri mereka dipahami. Selama hidup, setiap kali seseorang harus menyelesaikan sesuatu, di sisi lain interaksi tersebut ada orang yang sangat pandai menafsirkan apa yang mereka maksud. Dan perangkat lunak tidak dapat melakukan itu. Situs web pertama cukup sederhana, tetapi seiring dengan perkembangannya, desainer terus mengalami masalah ini, yaitu pengguna melakukan hal-hal yang tidak mereka antisipasi. Sementara itu, kami semakin ahli dalam mendesain untuk orang sungguhan.

Desain Antarmuka Pengguna

Seiring berjalannya waktu, jutaan, kemudian miliaran orang yang mengeklik produk dan situs web mulai menghasilkan data dalam jumlah besar. Faktanya, karena data ini sangat acak dan sulit diprediksi, maka diperlukan kelas alat yang sama sekali baru. Alat-alat ini menjadi "big data," yang berarti data yang tidak diklasifikasikan ke dalam baris dan kolom yang rapi, tetapi hanya diberi tag untuk dianalisis nanti. Big data adalah cara kita menangani aliran klik dan gesekan dan penekanan tombol dan waktu menonton video yang tidak dapat diprediksi dan seterusnya. Kita hanya membiarkan pengguna melakukan apa yang mereka inginkan, merekamnya, dan menyimpannya nanti.

Big data memungkinkan perancang antarmuka pengguna untuk mengumpulkan data yang tidak terduga dan melihat pola yang tidak mereka cari secara khusus. Sebaliknya, format data lama mengasumsikan Anda tahu apa yang akan Anda dapatkan – jadi mereka memiliki baris dan kolom yang rapi, seperti rekening bank atau nilai Anda di sekolah. Jumlah data yang sangat besar ini membutuhkan cara baru dalam melihat data, dan dua strategi digunakan untuk menangani semua kompleksitas baru dan data perilaku pengguna ini: standar desain dan pengujian. Standar desain sebenarnya hanyalah pelajaran yang dipelajari, sering kali dikombinasikan dengan beberapa panduan gaya visual. Misalnya, kami telah mempelajari bahwa sebagian besar pengguna mulai melihat situs web atau antarmuka produk apa pun di bagian tengah halaman, lalu ke kiri atas. Ini telah tertanam dari membaca makalah, dan masih berfungsi. Kami tahu bahwa orang tertarik pada gerakan di halaman, dan halaman yang berantakan membuat orang merasa stres. Ada banyak hal lain yang diketahui oleh desainer UI berpengalaman, dan mereka menciptakan titik awal untuk desain produk atau situs web apa pun. Agar berfungsi, antarmuka perangkat lunak harus memiliki data yang dibuat yang perlu dikirim ke perangkat lunak itu sendiri, sehingga perangkat lunak dapat memahami apa yang Anda perintahkan untuk dilakukan dengan mengetik, mengklik, atau cara apa pun Anda berinteraksi. Data tersebut mungkin disimpan secara permanen, atau mungkin hanya digunakan, lalu hilang. Ketika perancang perangkat lunak ingin memahami seberapa baik desain mereka menjalankan tugasnya, mereka mengumpulkan data ini, sering kali dalam jumlah besar. Dalam contoh ekstrem, perusahaan teknologi besar seperti Amazon dan Yahoo diketahui menjalankan ribuan eksperimen UI setiap hari. Mereka memindahkan tombol, mengubah deskripsi, mengubah beberapa warna, dan sebagainya. Semua ini dilakukan untuk membuat pengalaman menjadi lebih baik, dan pada akhirnya untuk memastikan lebih banyak pengguna membeli atau mengeklik iklan.

Sebagai pengguna perangkat lunak, penting bagi Anda untuk memahami bahwa sebagian besar antarmuka pengguna yang baik telah diuji secara ekstensif. Untuk perangkat lunak perusahaan, terutama perangkat lunak baru dari perusahaan muda, pengujian mungkin jauh lebih terbatas dan lebih seperti serangkaian wawancara, dan apa yang kami sebut pengujian "alfa", di mana sejumlah kecil pengguna mendapatkan versi awal perangkat lunak untuk mencobanya dan memberikan umpan balik. Nantinya, akan ada pengujian "beta", yang bertujuan untuk menyempurnakan antarmuka dan keandalan perangkat lunak.

Perusahaan akan terus melacak selamanya, berharap untuk terus menyempurnakan antarmuka dan memperkenalkan fitur dan cara baru untuk berinteraksi. Anda sering kali dapat memilih untuk tidak mengikuti pelacakan ini, biasanya di bagian "preferensi" di menu "pengaturan". Pada saat yang sama, Anda juga sering kali dapat menghubungi pembuat perangkat lunak yang Anda andalkan dan memberi tahu mereka apa yang ingin Anda lihat, dan apa yang tidak berfungsi. Tidak semua perusahaan siap untuk menangani hal ini, tetapi perusahaan yang baik selalu mencari masukan dan cara untuk memenuhi kebutuhan pengguna mereka dengan lebih baik. Saya akan memberi tahu Anda bahwa setiap perusahaan rintisan perangkat lunak harus bersemangat untuk mendapatkan masukan dalam bentuk apa pun, terutama dari pengguna yang belum mereka kenal.

Jenis Antarmuka Pengguna

Kita telah menempuh perjalanan panjang dari kartu berlubang. Saat ini, Anda dapat berinteraksi dengan perangkat lunak melalui pengetikan, klik, seret, sentuh, jepit, gerakkan ponsel, bicara, dan banyak lagi. Opsi antarmuka ini seharusnya membuat segalanya lebih jelas dan mudah digunakan. Sering kali memang demikian. Namun, pengalaman menggunakan beberapa antarmuka pengguna ini tidak selalu seperti yang biasa kita lihat pada antarmuka lama yang lebih berkembang. Misalnya, sebagian besar antarmuka pengguna grafis cukup fungsional dan dapat melakukan sebagian besar hal yang Anda inginkan dengan mudah, sementara ide-ide baru seperti antarmuka suara masih perlu dikembangkan.

Suara telah ada selama bertahun-tahun, tetapi baru-baru ini ia menjadi sesuatu selain pengambil catatan yang cukup rewel. Antarmuka suara yang benar-benar berguna dimulai dengan produk-produk seperti Siri Apple, dan dirilis sedikit sebelum teknologi untuk melakukan pekerjaan dengan baik cukup siap. Anda yang mulai menggunakan Siri ketika diperkenalkan pada tahun 2011 akan setuju bahwa itu adalah awal yang sulit. Namun, dalam beberapa tahun terakhir, kemajuan dalam kecerdasan buatan dan serangkaian pengujian pengguna telah membuat antarmuka pengguna suara jauh lebih intuitif dan canggih.

Demikian pula, siapa pun yang telah mencoba antarmuka realitas tertambah pada Hololens atau perangkat serupa akan setuju bahwa menyelesaikan sesuatu dalam AR masih memerlukan waktu untuk membiasakan diri. Teknologi yang mendasarinya cukup rumit, tetapi kita tidak terbiasa dengan "mengklik" di udara, dan itu terlihat jelas.

Saya pribadi berpikir bahwa realitas tertambah akan menggunakan suara lebih banyak daripada produk lain, tetapi kita lihat saja nanti. Tata letak dan pengoperasian antarmuka pengguna adalah hal pertama yang Anda pelajari dengan perangkat lunak baru, dan kemungkinan besar itulah sumber kebingungan awal yang mungkin Anda alami. Antarmuka itu dirancang untuk membuat hidup Anda lebih mudah, jadi jika pada awalnya Anda tidak memahaminya, buka YouTube dan cari tahu. Pengalaman saya adalah bahwa begitu orang mengetahui apa yang ada dalam pikiran perancang perangkat lunak, proses untuk mengetahui sisanya akan cepat.

2.5 PENGALAMAN PENGGUNA (UX)

Antarmuka pengguna adalah bagian yang paling jelas dari keseluruhan pengalaman menggunakan perangkat lunak untuk menyelesaikan pekerjaan. Kami menyebut pengalaman yang lebih luas itu sebagai "pengalaman pengguna," atau UX. Ini adalah pengalaman total yang Anda miliki sebagai pengguna perangkat lunak, dan berkaitan dengan hal-hal seperti urutan tindakan, pilihan fungsi mana yang akan ditempatkan, dan seterusnya. Perbedaan antara antarmuka pengguna (UI) dan pengalaman pengguna (UX) membingungkan, tetapi Anda dapat menganggapnya sebagai apa yang Anda lihat (UI) dan apa yang Anda lakukan (UX) dengan perangkat lunak.

Pengalaman menggunakan perangkat lunak dimulai dari pertama kali Anda memutuskan untuk mencobanya, hingga mempelajari cara menggunakannya, hingga penggunaan perangkat lunak yang sebenarnya, terutama saat Anda menggunakan perangkat lunak itu dengan perangkat lunak lain.

Perusahaan yang berbeda memilih untuk melihat UX dengan cara yang berbeda, dengan beberapa berusaha membuat pengalaman itu sangat positif, sering kali mencari "kesenangan" di antara pengguna; ini biasanya adalah perusahaan perangkat lunak konsumen seperti Apple atau Facebook. Perusahaan lain jauh lebih keras kepala dan ingin membuat pengalaman menjadi efisien, mudah dipahami, dan bermanfaat. Tidak mengherankan, ini adalah pendekatan yang dipilih banyak perusahaan perangkat lunak perusahaan.

Pengguna perangkat lunak konsumen juga merupakan pengguna perangkat lunak perusahaan, sehingga kebutuhan untuk memiliki UX yang dirancang dengan baik dengan sedikit kepribadian dapat juga mengalir ke perangkat lunak perusahaan. Kepala sekolah senior yang masuk ke iPad-nya untuk melihat gambar terbaru juga masuk ke Facebook untuk melihat karyawisata putrinya. Sangat sering, UX perusahaan dinilai berdasarkan UX konsumen, dan karenanya memiliki pengaruh.

Sama halnya dengan antarmuka pengguna, perusahaan perangkat lunak yang baik ingin mendengar pengalaman Anda, apakah Anda dapat berpindah dari satu area ke area lain, apakah Anda memahami langkah mana yang harus dilakukan dan kapan, dan seberapa baik perangkat lunak tersebut memulihkan kesalahan yang Anda buat atau orang lain buat, karena hal ini sering terjadi.

Sebagai pengguna, dan terkadang pembeli, perangkat lunak, saya tidak menyarankan Anda untuk bergantung pada UI, atau desain perangkat lunak, untuk membuat penilaian tentang apakah perangkat lunak tersebut tepat untuk Anda. Yang jauh lebih penting adalah pengalaman pengguna, terutama seberapa baik menu dan grafik dipikirkan, seberapa baik sistem terhubung di berbagai komponen, dan seberapa cerdas langkah-langkah untuk menggunakannya dipikirkan.

Saat Anda melihat perangkat lunak, perusahaan akan melakukan demo, dan tentu saja akan berfungsi dengan mudah. Saat Anda mencobanya sendiri, Anda seharusnya akan sedikit bingung pada awalnya, tetapi kemudian lihat berapa lama waktu yang dibutuhkan untuk mengetahuinya, dan lihat seberapa bermanfaat video orientasi perangkat lunak atau tutorial berbasis web. Selalu ingat agar perangkat lunak baru ini dapat berfungsi dalam skala apa pun,

banyak orang perlu mempelajarinya dengan cukup cepat. Jadi kurva pembelajarannya harus cepat, dan perusahaan perangkat lunak harus memiliki banyak materi untuk membantu.

Meskipun kebutuhan yang jelas untuk membuat orientasi menjadi lancar dan mudah, banyak perusahaan yang kurang membantu pengguna baru mempelajari cara menggunakan perangkat lunak mereka, di luar beberapa fungsi tingkat tinggi seperti login. Untuk perangkat lunak yang lebih mapan, pengguna lain akan menjadi sumber terbaik Anda untuk mengetahui cara membuat perangkat lunak tersebut melakukan apa yang Anda inginkan, dan sampai batas tertentu, untuk memahami seberapa bagusnya perangkat lunak tersebut, meskipun saya pikir sebagian besar orang akan setuju bahwa internet bukanlah sumber opini yang tidak bias.

Pengecualian, dan contoh bagus dari orientasi yang dipikirkan dengan matang adalah program "sertifikasi" Procore, di mana Anda mengikuti sekitar 90 menit tutorial video yang tidak dimaksudkan untuk menjadi sulit, tetapi menyeluruh dalam membahas setiap fungsi yang akan Anda perlukan. Procore memberi insentif kepada pengguna untuk melakukan tutorial ini dengan memberikan lencana yang dapat ditampilkan di LinkedIn. Jadi, dengan taktik ini, mereka berhasil memberikan pelatihan mendalam kepada pengguna dan juga membuat orang-orang membanggakan dukungan mereka terhadap Procore.

2.6 UJI COBA PERANGKAT LUNAK

Sejauh ini semua yang telah kita bahas adalah tentang betapa mudahnya perangkat lunak digunakan, tetapi pada akhirnya Anda membeli perangkat lunak agar perangkat lunak tersebut akan memberikan dampak menguntungkan bagi bisnis Anda yang cukup besar sehingga sepadan dengan kesulitan dan biaya penerapan yang sebenarnya. Satu-satunya cara untuk mengetahuinya adalah dengan uji coba.

Banyak perusahaan memiliki departemen inovasi yang dapat mendanai program uji coba formal, yang sangat penting. Tim inovasi harus mengembangkan kriteria untuk menilai perangkat lunak baru, mulai dari orientasi hingga dukungan pengguna hingga efektivitas aktual di tempat kerja. Jika memungkinkan, lebih baik menjalankan uji coba, bahkan untuk produk perangkat lunak yang sudah mapan, karena setiap produk perangkat lunak akan mengubah proses Anda – memahami bahwa dampak itu penting. Dampak ini sering kali didorong oleh pengalaman pengguna, yang pada gilirannya memungkinkan, atau mencegah, kru menggunakan produk tersebut dengan sukses.

Area penting di mana pengalaman pengguna terus berkembang adalah seberapa intuitif perangkat lunak tersebut merespons, dan bahkan mengantisipasi, apa yang akan dibutuhkan pengguna selanjutnya. Perangkat lunak yang mapan, seperti Autodesk dan Procore, biasanya lebih baik dalam hal ini, karena mereka punya waktu untuk membangun semua menu, opsi, dan fungsi tambahan yang memastikan hampir tidak ada pengguna yang terjebak, tidak dapat melakukan sesuatu yang penting untuk menyelesaikan pekerjaan.

Kita mulai melihat semakin banyak interaksi alami melalui suara yang memanfaatkan kecerdasan buatan. Pengalaman pengguna ini baru mulai mencapai apa yang saya yakini sebagai tingkat kemudahan dan efisiensi yang transformatif bagi pengguna.

Namun, karena suara membuat pengguna berasumsi bahwa mereka benar-benar berbicara dengan rekan yang mirip manusia, kita kembali melihat frustrasi pengguna karena perangkat lunak tidak selalu dapat memahami apa yang dikatakan pengguna, terutama ketika mereka menggunakan idiom yang tidak umum atau mengajukan pertanyaan tak terduga yang dapat ditangani dengan mudah oleh manusia. Yang mendasari semua produk ini adalah sistem AI berbasis data yang belajar dari waktu ke waktu, jadi seperti Siri saat ini yang jauh lebih baik daripada Siri tahun 2012, antarmuka suara untuk perangkat lunak konstruksi akan segera membuat perbedaan besar dalam kemampuan Anda untuk menggunakannya secara efisien, terutama di lapangan.

Aplikasi

Antarmuka pengguna adalah cara Anda memberi tahu perangkat lunak apa yang Anda inginkan, sedangkan pengalaman pengguna adalah cara perangkat lunak tersebut bekerja untuk Anda dalam menyelesaikan pekerjaan. Perangkat lunak itu sendiri disebut aplikasi.

Hingga pertengahan tahun 2000-an, sebagian besar aplikasi perangkat lunak dikirimkan kepada pengguna melalui beberapa jenis media penyimpanan, seperti disket atau CD-ROM. Kita mudah lupa bahwa pada tahun 2004 terdapat puluhan juta pengguna di AS yang terhubung ke internet melalui modem dial-up, jadi internet bukanlah cara yang baik untuk mendistribusikan perangkat lunak. Sebagai perbandingan, modem dial-up memiliki kecepatan tertinggi 56 kB per detik, dibandingkan dengan kecepatan Wi-Fi modern yang mencapai 2 MB/s. Banyak aplikasi yang Anda gunakan memulai kehidupannya di lingkungan internet 35 kali lebih lambat daripada yang paling lambat saat ini.

Seiring dengan semakin lazimnya penggunaan pita lebar, jenis aplikasi perangkat lunak baru mulai berkembang. Alih-alih mengirimkan perangkat lunak ke perusahaan yang kemudian mereka instal di komputer mereka sendiri, atau "di tempat," perangkat lunak mulai dihosting di server jarak jauh. Server tersebut biasanya ditempatkan di pusat data, bersama dengan ribuan server lainnya, dan dapat "disewa," jadi Anda tidak perlu memiliki semua peralatan itu sendiri. Kumpulan server ini menjadi apa yang sekarang kita kenal sebagai "awan." Awan memungkinkan perusahaan untuk menempatkan aplikasi di server yang berada di tempat lain, yang kemudian dapat diakses dari jarak jauh; dengan demikian, aplikasi tersebut dapat dijual bukan sebagai satu barang berwujud yang besar, tetapi sebagai langganan.

Langganan itulah yang kami sebut "Perangkat Lunak sebagai Layanan," atau "SaaS," dan telah mengubah lanskap teknologi dalam beberapa cara yang sangat penting.

Revolusi SaaS

Perangkat lunak dulunya cukup sulit dijual secara à la carte – Anda membeli semuanya, dan mungkin Anda membayar untuk "kursi", tetapi sebagian besar waktu Anda cukup membelinya, membuat satu pengguna, lalu dapat menggunakan semua yang dapat dilakukan perangkat lunak tersebut. Hal ini mempersulit perusahaan kecil untuk menggunakan perangkat lunak kelas perusahaan, karena banyak biaya yang harus dibayar di muka. Contohnya adalah Adobe Photoshop, perangkat lunak desain yang sangat umum untuk grafis. Pada tahun 2010, misalnya, Photoshop seharga $700 sebagai produk perangkat lunak mandiri.

Sekarang, Anda dapat mengakses Photoshop dengan harga di bawah $35 per bulan untuk satu pengguna. Itu adalah 5% dari harga untuk memulai, meskipun pada akhirnya Anda akan membayar lebih banyak jika terus menggunakan dan membayar bulanan. Banyak perusahaan dan pengguna yang tidak mampu membayar biaya satu kali dapat membayar bulanan. Karena biaya bulanan bertambah lebih banyak daripada biaya satu kali, perusahaan perangkat lunak mampu mengeluarkan lebih banyak biaya untuk pengembangan produk, yang dilakukan oleh sebagian besar perusahaan.

Selain biaya, perangkat lunak yang dikirimkan dari server perusahaan perangkat lunak berarti masih ada di server tersebut, dan itu berarti perusahaan perangkat lunak dapat membuat perubahan dengan mudah. Di masa lalu, setiap perubahan dimasukkan ke dalam "rilis" baru, dan itu terjadi cukup jarang, biasanya setahun sekali atau kurang. Di dunia saat ini, perusahaan terus-menerus merilis fitur baru, dan mampu mengubah UI dan UX dengan cepat saat mereka belajar dari pengguna. Namun, ini juga berarti bahwa jauh lebih mudah bagi pengguna untuk beralih ke produk perangkat lunak pesaing daripada sebelumnya, meskipun masih sulit untuk mengurai proses dan data.

Ketika perangkat lunak dikirimkan melalui internet sebagai layanan (SaaS), perusahaan perangkat lunak dapat memberi Anda kontrol yang jauh lebih besar atas seberapa banyak yang Anda gunakan, siapa yang dapat menggunakannya, dan banyak faktor biaya lainnya.

SaaS juga berarti bahwa aplikasi yang sama, dan dokumen serta data yang sama, dapat diakses di komputer mana pun dan sangat sering, tablet dan ponsel.

Aplikasi yang Diunduh

Meskipun SaaS sangat canggih, satu masalah yang selalu ada adalah konektivitas internet. SaaS biasanya memerlukan koneksi yang cukup baik, dan koneksi ini tidak selalu tersedia di luar kantor dan rumah. Akibatnya, aplikasi seluler, atau "aplikasi" masih bergantung pada versi yang diunduh untuk sebagian besar produk perangkat lunak. Misalnya, bukan hal yang aneh bagi produk SaaS yang tidak memerlukan unduhan untuk desktop atau laptop untuk memiliki aplikasi yang dapat diunduh untuk ponsel atau tablet.

Logika yang sama ini dapat diterapkan pada aplikasi yang memiliki persyaratan komputasi yang besar, terutama untuk grafik. Misalnya, sebagian besar perangkat lunak desain memiliki beberapa versi yang dapat diunduh, karena grafik umumnya membutuhkan banyak daya komputer, dan itu paling baik dilakukan secara lokal di komputer pengguna daripada dikirim melalui web. Anda juga akan melihat ini untuk realitas virtual, realitas tertambah, dan banyak aplikasi perangkat lunak kecerdasan buatan karena alasan yang sama. Faktanya, salah satu alasan mengapa jaringan nirkabel 5G begitu diminati adalah harapan bahwa sebagian dari pekerjaan berat ini juga dapat dialihkan dan diselesaikan dengan cara SaaS yang lebih murah dan terpusat. Hingga tulisan ini dibuat pada tahun 2020, 5G belum berhasil melewati beberapa kota uji coba, dan waktu peluncuran nasional dengan jangkauan yang sangat baik masih belum pasti.

Keamanan

Aplikasi juga merupakan pintu masuk ke seluruh dunia, dan tempat pelanggaran keamanan biasanya terjadi. Cara kerja perangkat lunak modern sangat saling terkait sehingga

terkadang sulit membayangkan bagaimana celah keamanan di satu tempat dapat memengaruhi basis data dan jaringan yang jauh, tetapi hal itu benar-benar dapat terjadi.

Beberapa pembaca mungkin ingat pada tahun 2018, ketika Target mencapai penyelesaian sebesar Rp. 185 Miliyar untuk pelanggaran data besar-besaran, di mana 40 juta catatan kartu kredit dicuri. Ternyata kredensial vendor HVAC mereka telah dicuri, yang kemudian memungkinkan peretas untuk masuk ke komputer utama Target dan mencuri data. Pelanggaran keamanan sulit dilaporkan, karena banyak yang tidak dilaporkan – tetapi dapat dipastikan bahwa setiap perusahaan yang Anda kenal, terutama yang besar, pernah mengalami pelanggaran. Yang paling umum adalah skema tebusan data, di mana peretas akan mengunci data perusahaan, dan meminta pembayaran untuk membuka kunci sistem.

Karena sebagian besar aplikasi terhubung satu sama lain melalui jaringan lokal atau internet, aplikasi yang Anda gunakan untuk hal-hal yang tampaknya tidak berbahaya seperti mengirim gambar ke anak-anak Anda dapat membahayakan seluruh perusahaan Anda, dan terkadang, klien Anda.

Meskipun ini bukan buku tentang keamanan, ini telah menjadi masalah besar sehingga setiap orang yang ingin memahami dan mengoperasikan teknologi konstruksi perlu memahami beberapa aturan utama:

1. Jangan pernah mengklik atau mengunduh apa pun dari email tanpa terlebih dahulu memeriksa apakah pengguna tersebut adalah orang yang disebutkan. Hal ini dapat dilakukan dengan mengklik nama pengguna di klien email Anda untuk memastikan tidak ada alamat email palsu.
2. Jangan pernah mengirim log-in atau kredensial lainnya melalui email, teks, atau cara lain. Jika Anda harus mengirimnya karena suatu alasan, pisahkan nama pengguna dan kata sandi, lalu kirimkan masing-masing dengan cara yang berbeda, seperti teks untuk yang satu dan email untuk yang lain. Cara terbaik adalah meletakkan kredensial dalam file Word, gunakan fungsi "Protect Document", tetapkan kata sandi, dan hubungi rekanan Anda dengan kata sandi tersebut.
3. Jangan pernah mengirim uang tanpa menelepon terlebih dahulu.
4. Anda dapat menemukan lebih banyak kiat seperti ini di situs web FBI. Karena banyak peretas di balik pelanggaran ini adalah orang asing, FBI mengerahkan sumber daya yang serius untuk membantu perusahaan-perusahaan kecil mengamankan sistem dan perilaku mereka.

2.7 SISTEM OPERASI

Aplikasi tidak bekerja sendiri, mereka memerlukan sesuatu yang akan menghubungkan banyak aplikasi ini ke perangkat keras yang menjalankannya. Tidak banyak yang perlu Anda ketahui, dan semuanya dapat dikelompokkan menjadi tiga jenis umum: komputer/server, seluler, dan sumber terbuka. Mari kita tinjau sekilas apa arti hal-hal ini, dan apa artinya bagi Anda.

Sistem Operasi Komputer

Semua orang mengenal Windows dan MacOS, keduanya adalah sistem operasi yang digunakan oleh 99% laptop dan desktop di dunia. Keduanya memberi tahu perangkat keras apa yang harus dilakukan, dan memungkinkan aplikasi untuk menggunakan perangkat keras, seperti speaker, memori, dan tentu saja, prosesor. Sebagian besar dari apa yang sebenarnya dilakukan oleh sistem operasi komputer cukup teknis. Windows dan MacOS melakukan banyak hal yang sama, tetapi memilih cara yang berbeda untuk melakukannya, dan dibuat dengan desain yang berbeda. Sistem operasi modern melakukan banyak hal yang dulunya merupakan aplikasi, seperti menghubungkan ke Wi-Fi, menjalankan pemeriksaan virus, dan banyak lagi. Jika Anda tidak benar-benar memahami sistem operasi selain hanya menemukan file, saya sangat menyarankan Anda meluangkan waktu sekitar satu jam untuk menonton video tutorial yang menunjukkan kiat dan trik, terutama tentang keamanan. Baik MacOS maupun Windows memiliki banyak fitur yang akan berguna bagi Anda – dan keduanya memiliki asisten yang diaktifkan dengan suara, yang merupakan cara gratis untuk mencoba teknologi baru ini.

Sistem Operasi Seluler

iOS Apple dan Android Google keduanya melakukan banyak hal yang sama – keduanya memungkinkan ponsel Anda menjadi komputer, mengoperasikan kamera, menghubungkannya ke internet, dan banyak lagi. Seiring dengan semakin canggihnya kamera, dan dengan penambahan radar versi ringan, yang dikenal sebagai LiDAR, ke iPad pada bulan Maret 2020, realitas tertambah akan menjadi kenyataan sehari-hari; iOS dan Android akan terus mengaktifkan aplikasi yang menggunakan fitur-fitur ini.

Sistem operasi seluler, khususnya Android, juga digunakan untuk menjalankan perangkat lain yang bersifat seluler dan memiliki beberapa karakteristik ponsel. Contoh yang bagus adalah Oculus Quest, yang merupakan headset realitas virtual yang dirilis pada tahun 2019. Sistem operasinya adalah versi Android, yang memungkinkannya memanfaatkan banyak aplikasi dan inovasi yang diperkenalkan Android.

Sumber Terbuka

Jenis sistem operasi ketiga sebenarnya tumpang tindih dengan dua jenis sistem operasi lainnya. Sementara Windows dan MacOS keduanya sepenuhnya dikendalikan oleh Microsoft dan Apple, telah terjadi pergerakan menuju perangkat lunak yang lebih terbuka dalam beberapa dekade terakhir. Misalnya, meskipun Android dimiliki oleh Google, standarnya jauh lebih fleksibel yang memungkinkan perusahaan dan kelompok lain membuat versi sistem operasi mereka sendiri, yang berguna saat Anda ingin menggunakannya untuk perangkat keras baru, seperti headset VR.

Mungkin sistem operasi sumber terbuka yang paling terkenal adalah Linux, yang sering digunakan untuk server dan kasus penggunaan yang lebih teknis, karena tidak dirancang atau didukung sebagai produk tingkat konsumen.

Stack

Kami mulai dengan ide yang paling berhadapan dengan pengguna, aplikasi. Dengan sistem operasi, kami telah bergerak lebih dekat ke perangkat keras yang sebenarnya, dan

sekarang saatnya untuk menggabungkan semuanya dengan konsep penting saat memikirkan teknologi apa pun – “Stack.” Kami menyebut berbagai hal sebagai tumpukan teknologi karena berbagai elemen bekerja pada tingkat yang berbeda, dari perangkat keras hingga pengguna.

Stack diperlukan karena pengguna tidak tahu cara berbicara dengan microchip, dan microchip memerlukan bahasa yang sangat spesifik untuk bekerja – yang kami sebut bahasa mesin atau biner. Jadi, kita perlu menerjemahkan dari satu lapisan ke lapisan lainnya, tetapi juga mempertahankan kemampuan untuk memiliki berbagai jenis aplikasi.

Di bagian bawah tumpukan adalah perangkat keras yang menjalankannya. Itu, pada gilirannya, biasanya berarti sebuah microchip dari beberapa jenis atau lainnya. Ini adalah keajaiban mutlak desain dan produksi, di mana miliaran transistor dikemas dalam ruang yang lebih kecil dari seperempat. Mari kita bahas apa itu transistor, dan apa fungsinya.

Transistor dan logika

Transistor hanyalah cara untuk menggunakan listrik untuk menyimpan nilai 1, atau 0. Ia melakukannya dengan menghubungkan kabel ke kabel lain, dan jika kabel pertama memicu sinyal listrik, ia akan menyebabkan kabel kedua membuka, atau menutup gerbang. Yang sebenarnya berarti adalah listrik dari kabel pertama membuat kabel kedua tidak dapat mentransmisikan listrik, atau mengubahnya dan membuatnya dapat menghantarkan listrik. Itu adalah angka 1, atau 0. Sekarang bayangkan sebuah chip dengan 1,7 miliar dari angka-angka tersebut, dan coba bayangkan bagaimana Mac atau PC Anda berubah dari Anda mengetik bahasa Inggris, menjadi entah bagaimana memerintahkan 1,7 miliar gerbang kecil tersebut untuk melakukan sesuatu yang berguna.

Hal semacam itu terjadi di semua perangkat lunak dan komunikasi, di mana terdapat lapisan perangkat lunak berbeda yang menerjemahkan dari apa yang dibutuhkan komputer untuk melakukan tugasnya, hingga apa yang dapat ditangani manusia, dan kemudian biasanya naik lagi ke apa yang paling sesuai untuk pekerjaan dan hiburan manusia yang sebenarnya.

Tumpukan biasanya dimulai dengan perangkat keras, yang dibuat untuk bekerja dengan sistem operasi tertentu. Sistem operasi tersebut kemudian memiliki aplikasi yang dapat Anda gunakan secara langsung, dan aplikasi ini terkadang mendukung aplikasi tingkat yang lebih tinggi – misalnya, browser web yang menyediakan Procore.

Tumpukan Pengembang Perangkat Lunak

Selain tumpukan pengguna yang baru saja kami jelaskan, ada penggunaan lain dari istilah "tumpukan" yang mungkin Anda temukan.

Ternyata aplikasi sering kali dapat dibuat dengan beberapa cara, menggunakan perangkat lunak yang berbeda. Mengingat bahwa tugas sebagian besar aplikasi perangkat lunak adalah menerjemahkan masukan satu tingkat ke atas atau ke bawah antara manusia dan perangkat keras, selama bertahun-tahun berbagai perusahaan dan pengembang telah menciptakan berbagai solusi untuk membuat aplikasi ini.

Bahasa – seperti C++, Java, Javascript, Python, PHP, atau Ruby – semuanya ada untuk membuat perangkat lunak yang akan bekerja dengan perangkat lunak lain untuk menyelesaikan berbagai hal.

Setiap kali perusahaan, atau pengembang, memutuskan untuk membuat aplikasi perangkat lunak, mereka akan memilih dari antara berbagai bahasa ini. Pilihan akan didasarkan pada apa yang dapat dilakukan oleh berbagai bahasa tersebut.

Bahasa-bahasa ini berfungsi karena menyertakan perangkat lunak mereka sendiri yang menggunakannya untuk membuat aplikasi Anda. Dengan kata lain, untuk membuat Java berfungsi, Anda perlu mengunduh paket Java ke komputer Anda, dan memuatnya agar berfungsi. Paket itu akan memungkinkan pengembang untuk menulis program dalam Java, kemudian pada akhirnya akan mengemas program tersebut, atau "mengompilasinya" menjadi aplikasi yang dapat Anda jalankan pada sistem operasi yang Anda buat. Beberapa bahasa yang lebih modern, atau versi bahasa, seperti Python 3, dapat bekerja tanpa dikompilasi, tetapi intinya adalah bahwa bahasa komputer adalah apa yang Anda gunakan untuk membuat aplikasi, melalui lingkungan pengembangan perangkat lunak mereka sendiri. Dan ini ada di level yang lebih rendah, seperti C++, di mana mereka berbicara dengan perangkat keras secara lebih langsung, atau level yang lebih tinggi, seperti Javascript, di mana mereka berbicara dengan aplikasi lain, seperti peramban web. Saat Anda ingin membuat perangkat lunak, atau bahkan mengerjakannya, mengetahui tumpukan perangkat lunak mana yang digunakan, dan apakah pengembang potensial Anda familier dengan tumpukan itu, sangatlah berguna. Seorang programmer yang baik dapat menggunakan bahasa apa pun, tetapi dalam praktiknya ada begitu banyak sumber daya yang ada, trik untuk menggunakan bahasa apa pun dengan baik, dan sebagainya, sehingga Anda ingin menggunakan programmer yang tepat untuk pekerjaan itu. Ini sangat mirip dengan pekerja terampil yang menjadi sangat ahli dalam jenis bangunan tertentu, seperti rumah sakit atau pusat data – Anda mempelajari begitu banyak hal spesifik tentang apa yang dibutuhkan jenis bangunan itu sehingga Anda jauh lebih produktif daripada orang lain yang berkualifikasi yang mungkin tidak memiliki pengalaman yang sama.

Dalam bab ini, kita telah membahas perangkat lunak, dari pengguna hingga perangkat keras. Setiap bagian dalam bab ini akan membantu Anda memahami cara kerja perangkat lunak dan apa yang dimaksud oleh beberapa spesialis yang akan Anda temui.

Dalam bab berikutnya, kita akan membahas perangkat lunak sebagai bagian dari jaringan. Karena sebagian besar perangkat lunak Anda adalah SaaS, menurut definisinya, perangkat lunak tersebut merupakan bagian dari jaringan.

BAB 3
JARINGAN PERANGKAT LUNAK

3.1 JENIS-JENIS JARINGAN

Perangkat lunak modern hampir selalu ada sebagai bagian dari suatu jaringan, sering kali lebih dari satu. Ada lima jenis jaringan komputer yang harus diketahui oleh sebagian besar pengguna, dan dipahami sampai tingkat tertentu: internet; Wi-Fi; seluler; satelit; dan jaringan API. Masing-masing bekerja secara berbeda, melakukan hal yang berbeda untuk Anda, dan memiliki risiko berbeda yang perlu dipahami.

Internet

Internet bukanlah hal yang baru, karena diciptakan oleh universitas untuk departemen pertahanan pada tahun 1960-an. Internet adalah standar terbuka utama, dan didasarkan pada beberapa aturan yang sangat longgar tentang cara memberi nama sesuatu, cara mengirim sesuatu dari satu titik ke titik lain, dan sistem untuk menemukan sesuatu di mana saja di dunia. Internet melakukan sebagian keajaiban ini melalui protokol internet (IP), yang memberikan "alamat" pada perangkat keras di jaringan. Faktanya, ini disebut "alamat IP," dan versi terbaru dari protokol internet terkadang disebut sebagai IPv6 (versi 6). Tidak ada yang aman secara inheren tentang internet, sama seperti tidak ada yang aman tentang jalan raya atau jalan samping. Jika Anda bukan pengembang perangkat lunak, tidak ada yang dapat Anda lihat tentang internet itu sendiri, karena tidak memiliki antarmuka grafis. Jadi kemungkinan Anda harus melakukan banyak hal dengan internet secara langsung cukup rendah.

Dalam Bab 2, kita membahas bagaimana internet menjadi dapat diakses di luar para insinyur perangkat lunak pada tahun 1994 dengan peluncuran, pertama, peramban Mosaic, kemudian Netscape. Perkakas perangkat lunak ini didasarkan pada inovasi dari seorang ilmuwan komputer Inggris, Tim Berners-Lee. Ia telah menciptakan sistem manajemen berkas yang disebut "protokol transfer hiperteks," yang membuat web berfungsi. Inilah sebabnya mengapa awal setiap alamat web lengkap memiliki http:// – yang merupakan singkatan dari protokol tersebut, atau protokol transfer hiperteks.

Anda juga akan melihat "https://" dan "s" merupakan singkatan dari secure, yang berarti bahwa informasi yang dikirim melalui situs web tersebut dienkripsi dan dikirim melalui apa yang mereka sebut secure socket. Aplikasi berbasis web apa pun yang Anda gunakan harus memiliki https:// di awal, atau pengembang harus menjelaskan mengapa mereka tidak mencantumkannya. Ada cara lain untuk mengamankan data, tetapi ini adalah yang paling mendasar dan umum.

Internet dan world wide web tidak bergantung pada satu jenis jaringan fisik. Semuanya berfungsi melalui seluler seperti halnya melalui koneksi kabel, atau Wi-Fi. Sebagian besar waktu Anda mengirim data, menerima email, menonton YouTube, atau terhubung ke perangkat lunak SaaS favorit Anda, Anda menggunakan http. Bahkan, saat Anda menggunakan perangkat lunak yang telah Anda unduh, perangkat lunak tersebut kemungkinan akan mengirimkan pesan kembali ke server penerbit untuk mengonfirmasi bahwa Anda memiliki

lisensi untuk menggunakan perangkat lunak tersebut. Hampir semua hal yang kita lakukan dengan cara tertentu bergantung pada internet untuk data, koneksi, atau kenyamanan.

Wi-Fi

Sebagian besar pengguna mengakses web melalui Wi-Fi, hampir sepanjang waktu. Wi-Fi lebih nyaman dan hampir tidak pernah ada masalah kinerja dibandingkan koneksi kabel. Beberapa tahun yang lalu, mengakses internet memerlukan kabel ethernet, dan banyak kantor masih memilikinya. Wi-Fi jauh lebih baik sehingga terkadang kita lupa bahwa kabel ini ada. Wi-Fi menggunakan serangkaian frekuensi radio tertentu untuk mengirim paket data berdasarkan protokol terpisah yang dikenal sebagai standar 802.11. Yang penting adalah bahwa seiring berjalannya waktu, Wi-Fi telah menjadi standar hampir di mana-mana, dan pada dasarnya tidak aman. Sinyal Wi-Fi dapat diamankan, dan sebagian besar perusahaan melakukannya di kantor mereka. Namun, kedai kopi, bandara, dan Wi-Fi "publik" lainnya tidak melakukannya.

Ada kemungkinan bahwa dalam beberapa tahun mendatang Wi-Fi akan menjadi usang karena permintaan bandwidth meningkat dan aplikasi internet of things (IoT) membutuhkan bandwidth yang semakin banyak. Saya berpendapat bahwa konektivitas 5G mungkin membuat Wi-Fi kurang diperlukan, tetapi kecepatan peluncuran 5G tidak pasti, dan pandemi tahun 2020 kemungkinan telah memperlambat segalanya lebih jauh.

Sementara itu, keamanan Wi-Fi dan terkadang kecepatan diatasi oleh perusahaan yang lebih besar dengan membuat apa yang disebut jaringan pribadi virtual (VPN). Ini tidak mahal lagi, dan sering kali berguna saat informasi sensitif dibuat atau dikirim. Jika Anda menjalankan perusahaan kecil dan ingin menjaga informasi klien Anda tetap aman, VPN bisa menjadi pilihan yang baik.

Namun, masalah besar untuk konstruksi adalah bahwa Wi-Fi sering kali sulit dipelihara di lokasi kerja yang cukup besar, membuat banyak jenis perangkat lunak menjadi kurang kuat, dan secara efektif tidak terhubung ke jaringan sampai dapat dihubungkan kembali di kantor.

Bluetooth

Bluetooth adalah protokol radio nirkabel perangkat ke perangkat yang dinamai menurut nama raja Denmark dari abad ke-10 yang menyatukan berbagai suku Denmark – idenya adalah Bluetooth seharusnya menyatukan berbagai perangkat dan aplikasi. Dan itu adalah cara yang bagus untuk melihat standar Bluetooth. Ini bukanlah cara yang bagus untuk menyelesaikan pekerjaan dengan bandwidth tinggi, itulah sebabnya kita masih memiliki Wi-Fi, tetapi ini adalah cara yang berguna untuk menghubungkan perangkat kecil ke perangkat yang lebih besar. Biasanya kita menghubungkan periferal, layar, dan earphone ke Bluetooth. Namun, terkadang ini bisa menjadi cara yang sangat mudah untuk menghubungkan perangkat yang berdekatan, dan Anda akan melihat hal semacam ini dari waktu ke waktu, terutama untuk sensor, seperti beacon yang digunakan untuk navigasi dalam ruangan. Bluetooth memiliki versi berdaya rendah, yang disebut "Bluetooth Low Energy" atau "BLE," yang sangat berguna untuk periferal yang memiliki baterai kecil.

Hal utama yang perlu diingat tentang Bluetooth adalah jaraknya pendek, dan fakta bahwa Bluetooth sangat sering diimplementasikan dengan cara yang bersifat hak milik –

sehingga perangkat Apple bekerja lebih baik dengan perangkat Apple lainnya melalui Bluetooth.

Seluler

Dalam beberapa tahun terakhir, jaringan seluler 4G cukup baik, dengan kesenjangan jangkauan yang lebih sedikit daripada sebelumnya. Untuk banyak aplikasi, jaringan ini berfungsi hampir sama baiknya dengan koneksi Wi-Fi yang baik. Banyak lokasi konstruksi yang tercakup dengan baik oleh jaringan 4G, sehingga jaringannya juga baik, meskipun masih ada lokasi yang terlalu terpencil – terutama pembangkit listrik, kincir angin, dan banyak lokasi pertambangan, minyak, dan gas.

Pada tahun 2020, pembicaraan tentang protokol berikutnya terus berlanjut, dan ada beberapa kota yang memiliki jangkauan 5G. 5G layak dijelaskan sedikit, karena pada suatu saat di tahun-tahun mendatang jaringan ini akan menjadi cukup berguna dan umum sehingga seorang teknolog konstruksi harus memahami apa yang seharusnya dapat dilakukan oleh jaringan ini.

5G didasarkan pada serangkaian frekuensi yang berbeda dari 4G dan sinyal seluler lainnya. 5G sendiri dapat berjalan pada berbagai frekuensi, tetapi semuanya merupakan frekuensi yang lebih tinggi daripada 4G, dan ini karena frekuensi yang lebih tinggi dapat membawa lebih banyak data, yang merupakan sebagian besar tujuan 5G.

Masalahnya adalah bahwa frekuensi yang lebih tinggi ini jauh lebih mudah diserap oleh dinding dan benda padat lainnya daripada 4G yang digantikannya. Jadi bagaimana tepatnya telekomunikasi akan meluncurkan 5G yang berfungsi di mana-mana masih harus dikerjakan. Kami memiliki banyak alasan untuk percaya bahwa hal itu akan dikerjakan, terutama melalui lebih banyak menara seluler daripada yang digunakan saat ini.

Selain dapat membawa hingga 100 kali lebih banyak data daripada 4G, manfaat utama lainnya dari 5G adalah waktu responsnya yang lebih cepat dibandingkan 4G, atau dikenal sebagai latensi yang lebih rendah. Latensi adalah waktu yang dibutuhkan sinyal untuk berpindah dari perangkat Anda, ke layanan atau perangkat lain, dan kembali lagi. 4G sekitar 50 milidetik, yang sangat cepat. 5G akan menjadi sekitar setengahnya, atau bahkan lebih cepat tergantung di mana Anda menerimanya.

Itu berarti bahwa aplikasi dengan bandwidth yang sangat tinggi dan respons cepat seperti realitas virtual dan desain dan konstruksi virtual (VDC) dapat dilakukan melalui internet, sedangkan saat ini kita perlu melakukan hal-hal ini dengan aplikasi yang diinstal secara lokal.

Kemungkinan besar 5G akan memakan waktu beberapa tahun untuk berfungsi sepenuhnya, karena implementasi awal teknologi seluler biasanya berada pada batas bawah dari apa yang secara teoritis dapat disediakan oleh teknologi tersebut, dan implementasi selanjutnya melihat teknologi tersebut semakin matang.

Jaringan Satelit

Banyak masalah yang mengganggu jaringan seluler mungkin dapat diselesaikan oleh banyak jaringan satelit orbit Bumi rendah yang dibangun oleh perusahaan-perusahaan seperti SpaceX dan Amazon. Ini akan memiliki ribuan satelit kecil yang cukup dekat dengan

permukaan Bumi untuk memiliki latensi rendah dan daya yang relatif tinggi. Saat artikel ini ditulis, teknologi ini masih dalam tahap pengiriman, dan sistem Starlink milik SpaceX diharapkan akan online pada akhir tahun 2020. Harga dan ketersediaan untuk digunakan oleh perusahaan konstruksi atau vendor perangkat lunak mereka belum ditetapkan, tetapi kemungkinan besar kemampuan baru ini akan mulai online pada tahun 2021. Sama seperti jaringan seluler, teknologi internet satelit kemungkinan akan memerlukan waktu untuk matang. Implementasi awal secara efektif akan menjadi "percontohan," karena jaringan menjadi semakin matang. Namun, jika aspirasi Starlink dan Amazon membuahkan hasil, mereka berjanji untuk mengubah cara dunia berinteraksi dengan informasi.

3.2 ANTARMUKA PEMROGRAMAN APLIKASI (API)

Semua jaringan lain sejauh ini berkaitan dengan transportasi, pada dasarnya merupakan jenis jalan yang berbeda. Dalam beberapa tahun terakhir, aplikasi mulai berbagi informasi secara langsung satu sama lain dengan cara yang sangat ampuh. Ada cara khusus agar aplikasi dapat berkomunikasi secara langsung satu sama lain. Dikenal sebagai antarmuka pemrograman aplikasi, metode komunikasi aplikasi ke aplikasi ini telah sepenuhnya mengubah cara perangkat lunak dibuat, dijual, dan direncanakan. Faktanya, API membentuk tulang punggung jaringan jenis baru, tempat aplikasi dapat berbagi data dan melakukan pekerjaan untuk satu sama lain dengan cara yang menciptakan pengalaman baru yang hebat bagi pengguna.

Kita dapat menganggap API sebagai jendela ke fungsi aplikasi perangkat lunak tertentu, tempat hal-hal yang melewati jendela itu berada dalam format yang mudah dipahami, yang memungkinkan aplikasi untuk melakukan tugas-tugas tertentu, lalu meneruskan informasi kembali melalui jendela.

Agar ini berfungsi, cara informasi dilewatkan melalui jendela dijabarkan dengan jelas, dan aplikasi atau pengguna yang meneruskan informasi harus menggunakan cara standar ini untuk melakukannya, dan akan dipersiapkan untuk data yang keluar sesuai dengan standar yang sama. Sebenarnya ada standar umum, yang disebut REST (REpresentational State Transfer) yang digunakan sebagian besar API. Karena standar ini sangat lazim, biaya dan kesulitan membuat API telah menurun drastis, sehingga API menjadi jauh lebih umum. Bayangkan sebuah jendela sebenarnya, mungkin di restoran drive-through. Untuk restoran ini, ada jendela input, tempat Anda memesan. Dan ada jendela output, tempat restoran menerima instruksi Anda, menindaklanjutinya, dan menyediakan apa yang Anda butuhkan. Dan seperti halnya API, ada hal-hal tertentu yang dapat Anda minta, misalnya burger dengan kentang goreng. Dan cara khusus untuk memintanya – jika Anda mengatakan saya ingin 100 gram daging sapi, dimasak di atas baja, dengan keju yang diiris tipis, di antara dua potong roti beragi, mungkin tidak akan dipahami. Namun jika Anda mengatakan “Quarter Pounder dengan Keju,” maka akan dipahami. Itulah yang dilakukan API – memastikan bahwa siapa pun dapat mengakses fungsi aplikasi tertentu jika mereka memiliki informasi yang tepat – dan informasi tersebut sering kali menyertakan kunci API, yang berfungsi seperti semua kunci lainnya (memungkinkan Anda masuk).

API digunakan sepanjang waktu – misalnya, saat Anda menggunakan akun Google atau Facebook untuk masuk ke situs web, situs web tersebut menggunakan API mereka untuk meneruskan permintaan autentikasi Anda, dan jika Anda memiliki akun dengan cookie di browser, Anda akan diautentikasi.

Dalam beberapa tahun terakhir, perusahaan seperti Salesforce telah memanfaatkan API untuk membuat platform yang memungkinkan perusahaan lain terhubung dengan mudah ke Salesforce, menarik data keluar, atau memasukkannya ke Salesforce, sehingga pengguna dapat memperoleh fungsionalitas yang berguna bagi mereka. Hal ini sangat sering berguna saat pengguna dan organisasi mereka memiliki kebutuhan atau minat khusus yang belum diputuskan untuk dikembangkan oleh platform. Misalnya, banyak perusahaan akan menghubungkan akun Salesforce mereka ke hal-hal seperti perangkat lunak akuntansi, atau perangkat lunak pembelajaran, dan banyak lainnya. Faktanya, saat artikel ini ditulis, Salesforce memiliki lebih dari 3.000 aplikasi pihak ketiga di bursa aplikasi mereka, dengan lebih dari 5 juta unduhan.

Kemampuan untuk menghubungkan satu sistem besar ke banyak solusi yang lebih kecil dan lebih sempit memiliki kekuatan yang sangat besar, dan sudah menjadi bagian penting dari cara perusahaan seperti Procore memasarkan produknya. Dalam beberapa tahun terakhir, Procore telah menciptakan App Marketplace, yang pada awal tahun 2020 memiliki lebih dari 220 aplikasi yang terhubung ke Procore dengan berbagai cara.

Misalnya, di pasar aplikasi mereka terdapat perusahaan bernama ArcGIS, dan seperti yang tersirat dari namanya, mereka adalah penampil untuk GIS, atau sistem informasi geografis. Layanan mereka terintegrasi dengan baik dengan kemampuan Procore yang sudah ada untuk menampilkan rencana dan gambar, dengan memungkinkan hal ini dipetakan dan dibandingkan dengan peta berkualitas tinggi. Dalam kata-kata dari situs web Procore:
Dengan pengalaman tertanam ArcGIS Viewer, Anda dapat:

- Mengintegrasikan aset akun ArcGIS Online Anda dengan proyek Procore Anda
- Melapisi foto proyek Procore pada Tampilan Peta ArcGIS Online Anda

Procore akan terintegrasi dengan semuanya mulai dari Structionsite untuk dokumentasi, hingga Microsoft Project, Oracle Primavera Suretrak, dan sekitar sepuluh aplikasi penjadwalan lainnya, dan seterusnya.

Setidaknya selama satu dekade terakhir, kekhawatiran yang terus berlanjut tentang penggunaan perangkat lunak konstruksi adalah bahwa berbagai hal tidak saling berhubungan. Hal itu masih berlaku untuk banyak produk perangkat lunak lama, tetapi perusahaan besar seperti Autodesk, Procore, MCiT, dan lainnya, semuanya berada dalam berbagai tahap dalam menciptakan platform yang terbuka dan mudah bagi mitra untuk terhubung dan memberikan nilai yang lancar kepada pelanggan mereka.

Cara kerja semua ini adalah karena platform ini menciptakan API, yang seringkali sangat lengkap. Melanjutkan contoh Procore, mereka menyediakan dokumentasi yang luas secara terbuka di situs web mereka, sehingga mitra dapat mulai membuat integrasi pada waktu mereka sendiri, kemudian bekerja dengan tim internal di Procore untuk menguji,

menyempurnakan, dan kemudian memasarkan integrasi tersebut di App Marketplace mereka.

Construction Cloud Connect dari Autodesk melakukan beberapa hal yang sama, seperti halnya CMiC dan lainnya. Mengingat keberhasilan perusahaan perangkat lunak di industri lain dengan pendekatan platform ini, dan tren umum yang menjadikan API sebagai hal mendasar dalam menjalankan bisnis, Anda dapat memperkirakan bahwa semakin banyak perusahaan perangkat lunak konstruksi akan saling terhubung dengan cara ini. Singkatnya, API adalah gerbang yang memungkinkan satu produk perangkat lunak untuk "meminjam" fungsionalitas dari produk perangkat lunak lain, dan dalam proses tersebut meneruskan data penting bolak-balik, untuk membuat penawaran yang lebih lengkap bagi pengguna.

Jaringan Teknologi

Gerbang-gerbang ini tidak hanya untuk laporan harian dan perangkat lunak desain. API adalah cara segala sesuatu mulai dari mesin hingga peralatan hingga manajemen konstruksi mulai dijalin menjadi jaringan produk di seluruh lokasi kerja yang saling berkomunikasi. Produk-produk ini dapat dikelompokkan menjadi tiga jenis: tablet dan ponsel, mesin, dan sensor. Dalam bab berikutnya, kita akan membahas lebih mendalam tentang sensor dan internet untuk segala hal; untuk saat ini, yang penting adalah bahwa ketiga produk ini terhubung melalui perangkat lunak, dan perangkat lunak itu sendiri menjadi semakin canggih sebagai hasil dari koneksi ini dan data yang mereka teruskan. Jaringan yang semakin saling terhubung ini pada gilirannya akan memungkinkan pengawas, manajer proyek, dan pemilik untuk lebih memahami lokasi kerja mereka, dan mengelolanya dengan jauh lebih efektif. Dalam sebagian besar proyek konstruksi saat ini, manajemen melihat informasi yang sudah ada sejak beberapa hari lalu, yang berarti bahwa setiap masalah sering kali ditangani dengan baik setelah masalah tersebut mulai memengaruhi pekerjaan. Nilai data, perangkat lunak, dan jaringan yang menghubungkannya adalah menjembatani kesenjangan waktu antara saat sesuatu terjadi dan kemampuan tim untuk mengatasinya. Namun, agar hal itu terjadi, pemahaman personel lapangan tentang nilai data sangatlah penting. Sebagai sebuah industri, konstruksi menghasilkan data dalam jumlah yang luar biasa, tetapi sangat sedikit yang digunakan. Faktanya, sebuah studi tahun 2018 oleh KPMG, sebuah firma akuntansi, menunjukkan bahwa lebih dari 95% data yang dibuat di lokasi kerja tidak digunakan dalam manajemen. Sebagian besar karena data tersebut hanya ada di selembar kertas, tetapi sebagian lagi karena data tidak dilaporkan selengkap atau seberguna yang seharusnya. Data Saat saya berbicara dengan para pemimpin senior di seluruh industri tentang teknologi konstruksi, salah satu area yang menjadi perhatian secara konsisten adalah bahwa orang-orang di lapangan tidak selalu memahami atau menghargai nilai data yang dikumpulkan, atau melihat mengapa data itu diperlukan. Bagaimanapun, sebagian besar tim lapangan ini adalah profesional berpengalaman yang benar-benar tahu apa yang mereka lakukan.

Yang lebih memprihatinkan adalah perasaan bahwa data sedang "dijadikan senjata" sehingga kontraktor umum memiliki cara lain untuk menekan subkontraktor. Kita mewarisi industri di mana beberapa ketentuan kontrak mendorong sikap yang berlawanan, dan kenyataannya adalah bahwa perselisihan adalah hal yang umum.

Namun, satu-satunya cara organisasi yang besar dan kompleks seperti tim konstruksi dapat meningkatkan kinerja dari waktu ke waktu adalah melalui pembandingan dan jenis analisis yang memerlukan data untuk dilakukan. Kebiasaan lama mengandalkan pengalaman pribadi, intuisi, dan "naluri" untuk membuat keputusan telah terbukti kurang efektif berulang kali, dan sedang dihapuskan di seluruh industri.

Pada saat yang sama, data dan perangkat lunak yang mengumpulkannya memberikan transparansi yang membantu semua orang. Sengketa dimenangkan atau dikalahkan berdasarkan dokumentasi, dan dokumentasi tersebut didukung oleh data.

Jadi, mari kita bahas tentang manfaat data, bagaimana data tersebut digunakan, dan bagaimana memastikan apa yang dikumpulkan dikumpulkan dengan baik dan tidak digunakan secara tidak tepat.

Mengapa Data?

Seorang pengawas, mandor, atau sekadar profesional berpengalaman di lokasi kerja akan dapat melihat bagaimana keadaan berlangsung pada hari tertentu. Mereka akan dapat melihat pola dalam cara kerja tim, menemukan masalah saat muncul, dan bertindak untuk mengatasi masalah tersebut. Itulah cara yang dilakukan sekarang, dan selalu dilakukan.

Tetapi apakah itu akan terus menjadi hal yang baik? Pertimbangkan betapa padat dan hampir tidak masuk akalnya hari seorang pengawas konstruksi sekarang. Banyak yang bangun pagi-pagi pukul 4 pagi, tiba di lokasi kerja pukul 6 pagi untuk melakukan inspeksi, dan pulang pukul 6 sore atau lebih, dengan sepanjang hari dihabiskan untuk menangani masalah, inspektur, pemilik, RFI, pengerjaan ulang, dan banyak lagi. Tuntutan akan kecepatan, keselamatan, perlindungan lingkungan, dan sekarang jarak sosial, terus meningkat, membuat tim di lapangan semakin sedikit waktu untuk bernapas.

Dan itu tidak lebih mudah bagi para pekerja, yang harus berpindah dari satu pekerjaan ke pekerjaan lain agar angka-angka mereka berhasil. Tidak ada waktu bagi kebanyakan orang untuk memikirkan tren jangka panjang, untuk melihat masalah rantai pasokan yang lebih besar daripada pekerjaan mereka, dan sebagainya. Setiap kontraktor perdagangan dan umum (GC) yang saya ajak bicara melihat perlunya memikirkan tentang bagaimana mereka melakukan apa yang mereka lakukan, untuk menganalisis apa yang berhasil dan apa yang perlu ditingkatkan, tetapi tidak ada dari mereka yang punya waktu untuk mengadakan rapat pasca-mortem, atau bertukar pikiran tentang cara kerja yang lebih efektif.

Perusahaan konstruksi dari semua ukuran ingin tumbuh dan menjadi lebih efisien, dan itu berarti bahwa manajemen perlu memahami status pekerjaan, tim, dan proses. Dan di luar sekadar pengoperasian bisnis, tanpa gambaran yang sangat jelas tentang berapa lama waktu yang dibutuhkan untuk menyelesaikannya sepenuhnya, estimasi menjadi permainan sandbag-making dan berasumsi Anda akan menutupi selisihnya dengan cara lain.

Rasa tidak punya waktu untuk berpikir ini tidak unik untuk konstruksi, dan di seluruh industri jawabannya selalu sama. Kumpulkan data di sepanjang jalan, dan gunakan untuk mempersingkat proses analisis proses. Faktanya, dengan menjadikan data sebagai bagian dari operasi sehari-hari, perusahaan mulai dari produsen hingga desainer hingga transportasi berat

semuanya telah menemukan cara untuk mempercepat perbaikan tanpa secara drastis meningkatkan waktu yang dihabiskan untuk rapat dan melakukan post-mortem.

Misalnya, apakah biaya sesuatu sesuai dengan yang Anda kira? Apakah proses membutuhkan jumlah jam kerja yang Anda kira? Biaya tenaga kerja adalah area yang paling sering salah perhitungan, karena kita cenderung memberi bobot lebih pada langkah-langkah penting seperti pengiriman, dan kurang memberi bobot pada langkah-langkah yang biasa saja dan terkadang menyakitkan seperti pengerjaan ulang dan jaminan kualitas.

Misalnya, saya mendapat kesempatan untuk berbicara dengan CJ Best, Direktur Manufaktur di McKinstry, kontraktor mekanik di Pacific Northwest. McKinstry dikenal sebagai perusahaan inovatif dengan fokus tajam pada manufaktur sebagai bagian dari cara mereka memecahkan masalah klien.

CJ memberi tahu saya bagaimana mereka menerapkan laporan harian dan pelacakan per jam untuk VDC mereka, dan dalam prosesnya memperoleh gambaran yang jauh lebih akurat tentang berapa banyak waktu dan upaya yang dibutuhkan untuk langkah-langkah tertentu dalam proses tersebut, dan jenis-jenis VDC tertentu. Inti dari latihan ini bukanlah untuk membuat alat untuk mendorong orang-orang mereka bekerja lebih keras. Sebaliknya, ini untuk dimasukkan ke dalam proses estimasi mereka untuk memperkirakan biaya, margin, dan risiko yang jauh lebih akurat untuk pekerjaan baru. Dalam prosesnya, CJ memberi tahu saya bahwa hal itu telah mengubah estimasi mereka secara dramatis untuk masa mendatang. Data seperti itu dapat dikumpulkan melalui perangkat lunak seperti Fieldwire, Fieldlens, dan tentu saja Procore, Autodesk, CMiC, dan lainnya. Meskipun sebagian besar lokasi kerja memiliki laporan harian dalam bentuk apa pun, laporan tersebut terlalu sering berupa kertas, atau penghitungan jam kerja yang sangat terbatas. Bagaimana jika Anda dapat mengumpulkan sedikit data, setiap hari tentang kinerja tim dan mencatat inefisiensi serta hambatan dalam menyelesaikan pekerjaan, dan sebagainya?

Cara Data Dimanfaatkan

Nilai sebenarnya dari laporan harian muncul saat laporan tersebut dikumpulkan dan dilihat bersama data lain, dan untuk melakukannya, sebagian besar perusahaan akan berintegrasi dengan perusahaan besar seperti Autodesk atau Procore. Produk perangkat lunak ini hadir dengan paket analisis mudah yang memungkinkan Anda mengajukan pertanyaan sederhana dan melihat jawabannya baik sebagai angka maupun sebagai grafik.

Faktanya, salah satu hal terpenting yang dapat dilakukan perangkat lunak untuk Anda adalah mengambil tumpukan besar data dan mengolahnya menjadi grafik yang mudah dipahami dan cepat ditafsirkan. Kekuatan data grafik dapat diabaikan, tetapi pertimbangkan ini – bagian terbesar dari otak Anda adalah korteks visual. Anda memiliki kemampuan lebih untuk menemukan pola dan melihat masalah melalui mata Anda daripada dengan cara lain. Hal ini berlaku bagi para ahli maupun pemula. Jadi, selalu minta data untuk dibuat grafik jika Anda memiliki kesempatan.

Tujuan pengumpulan data adalah agar seluruh operasi dapat dijalankan dengan lebih aman dan efisien. Kita lebih mampu melakukan ini berdasarkan data, karena tidak ada satu orang pun yang melihat semuanya, pada setiap tingkat detail, dan mengingatnya dengan

sempurna. Penting untuk memisahkan apa yang menjadi keahlian orang, dan apa yang menjadi keahlian alat kita. Komputer memiliki memori yang sempurna, pencarian yang sangat cepat, dan kemampuan untuk menyajikan analisis kepada manusia. Namun, mereka sama sekali tidak memahami apa artinya semua itu. Pekerjaan manusia selalu menerapkan pengalaman dan penilaian pada pekerjaan, dan itu tidak berubah. Data tidak membuat keputusan, data tidak menggantikan nilai "perasaan", data sekarang menjadi alat untuk mendukung intuisi manajer berpengalaman di mana pun.

Pertentangan antara teknologi dan pengambilan keputusan berdasarkan data di satu sisi, dan pengalaman, penilaian, dan intuisi profesional di sisi lain, adalah keliru. Alih-alih membiarkan komputer menggantikan penilaian manusia, justru ketika Anda memiliki perangkat lunak yang dapat menarik data, menganalisis data tersebut, dan kemudian menunjukkan hasil kepada Anda, Anda memerlukan seorang profesional untuk memahaminya. Nancy Novak, kepala bagian inovasi Compass Datacenters, menceritakan bagaimana ia selalu melatih timnya untuk melihat data sebagai titik awal. Setiap bagan dan analisis data yang tampak jelas dan meyakinkan perlu diperiksa ulang, karena datanya bisa sangat meleset. Sering kali, data mengungkapkan fakta dan tren yang tidak akan pernah dilihat tim dalam kegiatan bisnis normal, tetapi bahkan saat itu pun apa arti fakta tersebut tidak pernah diungkapkan oleh perangkat lunak – terserah kepada para profesional konstruksi untuk memahaminya.

Itulah poin umum tentang teknologi konstruksi – teknologi tidak menggantikan apa yang dapat dilakukan manusia, tetapi melengkapi apa yang dapat dilakukan manusia. Dan dalam prosesnya, kita sebagai individu, perusahaan, dan industri dapat melakukan hal-hal yang tidak akan pernah dapat kita lakukan tanpa bantuan teknologi tersebut.

Tidak seorang pun membantah bahwa memiliki forklift lebih baik daripada 10 orang yang berlarian ke sana kemari mencoba membawa barang-barang di sekitar lokasi kerja. Tidak seorang pun membantah bahwa memiliki derek memungkinkan untuk membangun gedung yang jauh lebih tinggi, lebih kuat, dan secara umum lebih baik daripada yang dapat kita lakukan jika kerekan dan katrol adalah satu-satunya cara kita dapat mengangkat barang, belum lagi peningkatan keselamatan yang dramatis.

Data melakukan hal yang sama, hanya dengan cara yang kurang terlihat. Saya suka membuat analogi bahwa data dan perangkat lunak seperti alat konstruksi dan barang habis pakainya. Tukang las membutuhkan kawat, pistol paku membutuhkan paku, mixer membutuhkan semen, dan seterusnya. Di dunia saat ini, pengawas dan manajer membutuhkan data.

Namun tidak seperti barang habis pakai lainnya, data tidak "habis" saat Anda menggunakannya. Anda dapat berbagi data secara internal dan dengan perusahaan lain dan data tersebut tidak akan habis, atau nilainya akan berkurang. Faktanya, karena kita menggunakan data untuk melihat hal-hal yang tidak dapat kita lihat dengan mata telanjang, sering kali data menjadi lebih berharga saat dibagikan.

Misalnya, mengetahui produktivitas tim Anda akan membantu Anda mengelolanya, tetapi tidak akan membantu Anda melihat seberapa baik kinerja Anda. Di sinilah

pembandingan menjadi penting – dengan berbagi data Anda dengan tim lain, atau sekelompok tim di seluruh perusahaan, Anda semua dapat melampaui sekadar mengelola tim Anda; sekarang Anda memiliki apa yang perlu Anda tingkatkan.

Sebelumnya, saya mengemukakan kekhawatiran yang sering muncul – bahwa data digunakan sebagai senjata melawan tim lapangan. Pembandingan dapat dengan mudah menjadi contohnya, jika di situlah Anda menghentikan percakapan. Bagaimana jika alih-alih hanya melihat seberapa banyak yang diselesaikan, data mencakup bagaimana hal-hal diselesaikan? Bagaimana jika rapat harian mencakup diskusi singkat tentang proses yang digunakan, masalah yang dihadapi, dan sebagainya.

Ini kemudian dapat dimasukkan ke dalam laporan harian atau perangkat lunak lain yang sudah ada sebelumnya untuk mengumpulkannya dari waktu ke waktu. Lalu apa yang tampak seperti ukuran hukuman, pembandingan, menjadi indikator bahwa beberapa tim telah menginovasi proses yang dapat dipelajari oleh tim lain. Dengan menggabungkan kinerja yang lebih tinggi dari tim lain dengan rincian tentang bagaimana mereka mencapainya, organisasi secara keseluruhan mendapat manfaat. Hal ini menghilangkan aspek persenjataan dengan mendasarkan perbandingan antartim pada informasi yang nyata dan dapat ditindaklanjuti.

Hal inilah yang dilakukan di industri mobil, dan merupakan salah satu fondasi gerakan Lean. Idenya adalah Anda tidak pernah berhenti pada hasil saja, tetapi Anda melihat proses yang sebenarnya dan terus mengajukan pertanyaan hingga Anda mendapatkan sesuatu yang benar-benar dapat Anda tingkatkan. Data adalah satu-satunya cara Anda dapat melakukannya dalam skala besar, berulang kali.

Tentu saja, cara lain untuk menghilangkan persenjataan data adalah dengan meluangkan waktu dan mempelajari data yang dikumpulkan, sistem yang digunakan (seperti Procore), dan mengajukan pertanyaan yang lebih baik tentangnya. Ada banyak orang di bidang perdagangan yang melakukan ini sekarang, karena pelatihan sering kali gratis dan tidak butuh waktu lama untuk menjadi ahli dalam hal ini sebagaimana yang Anda perlukan.

Agar bernilai, data perlu dilihat sebagai sumber pertanyaan yang lebih baik dan beberapa jawaban, tetapi jarang sekali merupakan jawaban yang sebenarnya. Itu membutuhkan penilaian dan pengalaman.

Pengumpulan data juga perlu dilihat sebagai bagian dari peningkatan proses, bukan negosiasi kontrak. Pengumpulan data memerlukan partisipasi yang sukarela dari tim di lapangan, dan begitu mereka melihat data tersebut sebagai upaya untuk mendapatkan keuntungan, kualitas dan keandalan data akan menurun, seperti yang tampaknya terjadi di banyak lokasi kerja saat ini.

3.3 MEMERIKSA DATA SECARA INTUISI

Sebagai seorang profesional dengan pengalaman bertahun-tahun, penilaian unik yang Anda berikan jauh lebih baik daripada buku seperti ini. Namun, kami dapat memperkenalkan beberapa alat dan aturan umum tentang data yang dapat membantu Anda menerapkan intuisi

dan menemukan data yang bermasalah. Dalam prosesnya, Anda juga akan menjadi pengelola data yang lebih baik yang Anda dan tim hasilkan.

Mari kita ulas kembali dan perjelas apa yang kami maksud dengan "data." Kata tersebut digunakan sepanjang waktu untuk mengartikan hal yang berbeda dan definisi prinsip dasar yang baik akan membantu.

"Data adalah kumpulan fakta dan pengamatan yang digunakan untuk operasi dan analisis."

Ada beberapa poin penting di sini.

Data Berkaitan dengan Melakukan atau Menganalisis Pekerjaan

Poin pertama adalah bahwa fakta apa pun tidak memenuhi syarat sebagai "data." Fakta menjadi data saat digunakan untuk melakukan pekerjaan atau memahami pekerjaan tersebut. Pikirkan tentang apa artinya dari dua perspektif, yakni membuat dan menggunakan data.

Dalam hal membuat data, penting bagi kita untuk memahami mengapa data dikumpulkan: Pekerjaan atau operasi apa yang kita harapkan akan berjalan lebih baik atau lebih dipahami? Dalam kebanyakan kasus, hal ini akan mudah diketahui karena data dikumpulkan dalam proses melakukan pekerjaan. Kita mengajukan pertanyaan secara eksplisit sehingga kita dapat dengan jelas memikirkan data apa yang penting dan data apa yang tidak akan membantu.

Sebagian besar data yang akan Anda buat secara pribadi adalah laporan tentang kemajuan, kualitas, dan tenaga kerja. Masing-masing data ini dapat dipalsukan, dapat dilaporkan dengan cara yang tidak jelas, atau bahkan dapat terlewatkan pada suatu waktu. Pada saat itu, mungkin tampak bukan masalah besar untuk melewatkan atau memalsukan data harian, tetapi ingatlah bahwa Anda sedang menciptakan sesuatu yang akan membantu Anda dan tim Anda pada waktunya. Data tersebut akan dikumpulkan dari waktu ke waktu dan membantu Anda menjalankan pekerjaan dengan lebih baik, mempertahankan diri dari klaim tertentu, dan secara umum menjelaskan kepada semua orang bagaimana keadaan sebenarnya, hari demi hari. Untuk melakukan semua ini, data perlu dianalisis sebagai kumpulan besar dari waktu ke waktu, dan analisis tersebut sangat bergantung pada apa yang Anda dan orang lain lakukan setiap hari.

Sebagian besar perusahaan pernah menghadapi masalah konsistensi data di beberapa titik. Faktanya, saat ini sebagian besar perusahaan masih menghadapi masalah ini. Karena konstruksi pada dasarnya adalah proses yang terfragmentasi, dengan kru yang berbeda pada pekerjaan yang berbeda di kota yang berbeda dengan PM yang berbeda, data sebelumnya biasanya dalam format yang dibutuhkan PM, bukan standar seluruh perusahaan.

Karena perusahaan telah mengadopsi perangkat lunak manajemen proyek modern, seperti Autodesk, CMiC, atau Procore, nilai dari melihat berbagai hal di seluruh perusahaan menjadi lebih jelas. Jaringan perangkat lunak ini membantu semua orang saat datanya jelas, konsisten, dan tepat waktu.

Nilai ini untuk membantu tim melakukan pekerjaan mereka tidak selalu jelas, tetapi saat data tersebut, dan cara memasukkannya, menjadi konsisten, cara-cara yang akan dilakukannya untuk membantu menjadi jelas.

Di Harkins Builders, kontraktor umum regional yang berkantor pusat di Wayne, Pennsylvania, Patrick Hennessy bekerja dengan tim internal untuk membuat cara laporan diajukan, masalah diklasifikasikan, dan secara umum cara data dimasukkan konsisten.

Ketika Patrick pertama kali bergabung dengan Harkins, ia menemukan bahwa proses dan data yang dihasilkan berbeda-beda dari satu tim ke tim lainnya. Pat dan timnya menemukan bahwa cara terbaik untuk mengatasi masalah ini adalah dengan membentuk tim khusus untuk menghubungi masing-masing pengguna perangkat lunak dalam organisasi mereka, dan menciptakan cara umum untuk memasukkan informasi, sehingga menciptakan cara yang sama sekali baru dalam memandang bisnis.

Dengan dukungan dari tingkat senior, Pat dan timnya mampu menyelaraskan banyak dari proses yang berbeda ini, sehingga manajemen di berbagai tingkatan kini memiliki alat yang ampuh untuk membantu tim agar berhasil baik secara langsung maupun dalam mempersiapkan diri dengan lebih baik untuk pekerjaan baru.

Salah satu cara Harkins memastikan bahwa mitra mereka dalam bidang perdagangan memahami bahwa data dan dasbor yang dibuatnya digunakan untuk keuntungan bersama, bukan hanya sebagai alat untuk menekan bidang perdagangan, adalah dengan menempatkan tim mereka sendiri dalam bagan dan analisis yang sama dengan mitra dagang mereka. Taktik sederhana ini menunjukkan banyak hal tentang pendekatan umum mereka – semua orang menghargai berada di “perahu yang sama,” perasaan yang tidak selalu lazim dalam hubungan dagang GC.

Pat membagikan beberapa cara data ini memungkinkan mereka mengelola pekerjaan dan proses dengan lebih efektif, dan contoh yang bagus adalah laporan risiko proyek mereka. Harkins kini memiliki laporan yang mencakup 10 faktor yang menunjukkan bahwa suatu proyek berisiko – hal-hal seperti persentase penyelesaian terhadap rencana, dan KPI lainnya. Sebelum mereka menyelaraskan data di seluruh perusahaan, manajemen hanya akan mengetahui suatu pekerjaan bermasalah saat kemungkinan besar akan gagal. Kini CEO dapat melihat dasbor yang berisi lebih dari 50 pekerjaan sekaligus, dan melihat secara langsung di mana sumber daya dibutuhkan untuk membantu tim yang tertinggal. Intinya bukanlah untuk bertindak gegabah, melainkan untuk memastikan proyek tersebut berhasil.

Mungkin yang terpenting, tim telah mampu menghubungkan 10 KPI ini dengan pekerjaan yang merugi. Dalam kata-kata Pat, “Kami selalu tahu saat pekerjaan gagal, kini kami tahu mengapa pekerjaan itu gagal.” Sekarang mereka dapat bertindak tepat waktu untuk menyelamatkan pekerjaan-pekerjaan ini.

Yang merupakan poin kedua kita: Data digunakan untuk pemahaman dan analisis. Selain sekadar menyelesaikan pekerjaan, memahami apa yang mendorong keberhasilan atau kegagalan selalu penting. Data memungkinkan para manajer untuk melihat faktor-faktor keberhasilan dan kegagalan ini dengan cara yang tidak mungkin dilakukan hanya dengan pengamatan manusia.

Analisis: Data + Penilaian

Tidak ada perangkat lunak yang dapat melakukan beberapa hal yang dianggap biasa oleh manusia. Perangkat lunak tidak dapat memahami konteks suatu situasi, tidak dapat memahami intuisi ketika ada sesuatu yang tidak beres. Kita terbiasa melihat perangkat lunak unggul dalam tugas-tugas yang sulit bagi manusia. Yang sebaliknya terjadi jauh lebih sering – hal-hal seperti penilaian, pemahaman yang kompleks, dan intuisi adalah hal-hal yang dapat dilakukan manusia dengan mudah. Perangkat lunak tidak dapat menangani apa pun di luar kasus-kasus sempit yang menjadi tujuannya.

Menganalisis data berarti menyatukan kedua kekuatan yang saling melengkapi ini. Perangkat lunak, dan data yang Anda masukkan ke dalamnya, dapat menunjukkan pola lintas waktu, lintas pekerjaan, dan lintas tim, yang tidak terlihat oleh manajer tanpa bantuan. Namun, pola-pola tersebut tidak secara otomatis memiliki makna apa pun, pola-pola tersebut mungkin hanya pola acak dalam data, atau mungkin disebabkan oleh sesuatu yang tidak relevan, seperti cuaca.

Dalam contoh Harkins, tidak ada seorang pun yang memiliki kemampuan untuk memahami apa yang terjadi di 50 pekerjaan, tanpa perangkat lunak dan data yang diformat secara konsisten untuk menyediakannya bagi mereka. Namun, setelah data dikumpulkan, pengamat data tersebut harus menerapkan penilaian mereka sendiri untuk memutuskan kapan data benar-benar menunjukkan suatu pekerjaan bermasalah, dan kapan tidak.

Karena manajer dapat mengenal pemimpin tim secara pribadi, ia dapat menyelidiki situasi dan mencari tahu apakah ada sesuatu yang terjadi di luar apa yang ditunjukkan data. Manajer yang sama, yang telah terlibat dalam konstruksi selama bertahun-tahun, juga akan dapat menafsirkan data dalam konteks fakta lain dan pemahaman umum tentang bagaimana pekerjaan berhasil atau gagal.

Kekuatan teknolog konstruksi modern berasal dari penerapan pengalaman konstruksi yang diperoleh dengan susah payah ini pada data dan analitik yang baru dibuat yang memungkinkan mereka melihat lebih dari sekadar pekerjaan mereka sendiri atau kelompok terdekat.

Standar Data

API menciptakan cara yang fantastis untuk menyatukan berbagai sistem perangkat lunak menjadi satu jaringan data dan fungsi yang berguna. Namun, API hanyalah jendela, saluran antara berbagai produk perangkat lunak. Dalam beberapa kasus, data yang dibuat oleh satu produk perangkat lunak berguna untuk produk lain. Namun, dalam kebanyakan kasus, API dari satu produk mengharuskan produk lain untuk mematuhi persyaratan spesifiknya agar bermanfaat, dan sebagian besar produk perangkat lunak hanya melakukan ini dengan beberapa produk lainnya.

Contoh arcGIS sebelumnya menggambarkan maksud saya. arcGIS menggunakan format standar untuk informasi geografisnya. Ini adalah format yang sudah digunakan Procore, karena merupakan format standar. Hal yang sama berlaku untuk banyak format file lainnya, seperti JPEG, atau MP4 – format standar ini berarti Anda dapat dengan mudah

berbagi foto dan video. Namun, standar data harus lebih dari sekadar format file, karena kita melakukan lebih dari sekadar menampilkan media dengan data.

Tujuan data adalah untuk mengelompokkannya bersama-sama, agar dapat menjalankan beberapa analisis terhadapnya, seperti rata-rata dan persentase perubahan, dan sebagainya. Untuk menjumlahkan semuanya, semuanya harus dalam format yang sama, dan ini hampir tidak pernah terjadi secara alami.

Ingat sebelumnya ketika kita membahas pembangun Harkins. Mereka harus memiliki tim senior yang menghabiskan waktu hampir dua tahun untuk membuat tim proyek mereka yang berbeda menggunakan KPI yang sama, menggunakan proses yang sama untuk memasukkannya ke dalam sistem. Mereka menghabiskan uang dan sumber daya untuk mengubah cara mereka menangani data.

Sekarang bayangkan Anda memiliki 50 perusahaan perangkat lunak manajemen proyek. Atau Anda memiliki perangkat lunak desain, perangkat lunak laporan harian, perangkat lunak manajemen keselamatan, dan sebagainya. Masing-masing perusahaan ini akan membangun produk mereka secara independen, dan akan membuat keputusan tentang cara memformat data mereka.

Akibatnya, perangkat lunak dari berbagai perusahaan sering kali tidak bekerja sama, terutama jika menyangkut data yang sangat terperinci tentang kinerja pekerjaan. Hal ini menyebabkan perusahaan, terutama kontraktor perdagangan khusus, memasukkan data yang sama dua kali atau bahkan tiga kali lipat ke dalam sistem yang berbeda.

Selama saya berkecimpung dalam teknologi konstruksi, ini adalah keluhan yang paling umum. Dua hal dapat dilakukan untuk mengatasi hal ini. Yang pertama terjadi cukup cepat, yaitu perluasan platform perangkat lunak besar, seperti Autodesk Construction Cloud dan Procore. Dengan menghadirkan semakin banyak fungsi ke dalam satu platform, masalah standar data menjadi tidak relevan. Akan tetapi, tidak ada satu perusahaan pun yang dapat menjadi yang terbaik dalam segala hal, khususnya dalam bidang perangkat lunak di mana ide-ide baru terus bermunculan.

Hal kedua yang dapat dilakukan untuk mendorong standarisasi data adalah apa yang terjadi di banyak industri lain – seseorang menciptakan sebuah standar. Dalam dunia perangkat lunak yang lebih luas, standar tersebut sering kali adalah IEEE (Institut Insinyur Listrik dan Elektronik) – mereka adalah kelompok yang menetapkan standar Wi-Fi, misalnya.

Karena konstruksi sangat terfragmentasi, tetapi juga terlalu sempit untuk kelompok seperti IEEE, sulit untuk mengembangkan standar data yang diadopsi secara luas. Ada satu standar yang muncul yang merupakan gagasan dari beberapa penggemar teknologi konstruksi awal. Disebut CDX, atau pertukaran data konstruksi, standar ini telah dibuat dengan susah payah oleh kelompok nirlaba yang disebut Construction Progress Coalition. CDX lebih banyak membahas pendekatan terhadap standarisasi data daripada sekadar standar yang sebenarnya, karena Nathan Wood dan timnya menyadari sejak awal pelajaran yang sama yang dipelajari Patrick di Harkins Builders – industri ini dipenuhi dengan pembangun yang sangat berpengalaman dan sangat profesional yang mengandalkan apa yang telah berhasil di masa lalu untuk memastikan mereka berhasil menyelesaikan proyek berikutnya. Pengalaman itu

biasanya menyebabkan keengganan untuk mencoba hal-hal baru yang mungkin menyebabkan kerumitan dan ruang untuk kesalahan di masa mendatang. Dan orang-orang yang sama itu terlalu sibuk untuk meluangkan waktu untuk memahami cara baru memasukkan data, atau mengajukan RFI, pengajuan, atau dokumen lainnya.

Saat ini CDX sedang bekerja keras untuk membuat standar RFI baru, dan keberhasilan mereka dalam mengajak orang-orang untuk hadir menunjukkan betapa pentingnya hal ini, tetapi fakta bahwa mereka belum berhasil dalam mengadopsi standar secara luas menunjukkan betapa sulitnya bagi standar eksternal untuk diadopsi oleh perusahaan-perusahaan perorangan, terutama perusahaan perangkat lunak yang sangat yakin perangkap tikus mereka lebih baik. Tentu saja, BIM adalah standar perangkat lunak utama, tetapi sebenarnya tidak ada "standar" melainkan kumpulan produk dan praktik yang termasuk dalam nama tersebut. Jika sesuatu yang mendasar seperti BIM membutuhkan waktu puluhan tahun untuk menjadi standar, kita dapat memahami betapa industri ini masih menolak standar lain.

3.4 PERANGKAT LUNAK SEBAGAI SISTEM

Dalam bab ini, kita telah mulai dari dasar untuk memahami dasar jaringan, cara kerjanya, API yang sangat penting, dan bagaimana data dapat digunakan untuk meningkatkan bisnis pembangunan gedung.

Masalah pemahaman data, dan cara menganalisisnya serta menggunakannya akan terus menjadi penting bagi siapa pun yang ingin menguasai teknologi konstruksi saat ini, dan di masa mendatang. Kami membahasnya di sini secara mendalam, tetapi ada banyak sumber daya gratis di luar sana yang menawarkan pemahaman yang lebih mendalam, salah satunya adalah kursus saya di procore.org.

BAB 4
PERANGKAT LUNAK KONSTRUKSI

Menurut data dari PMI-Plangrid, “35% waktu profesional konstruksi dihabiskan (lebih dari 14 jam per minggu) untuk aktivitas yang tidak produktif, termasuk mencari informasi proyek, penyelesaian konflik, dan menangani kesalahan serta pengerjaan ulang.” Studi tahun 2018 ini menemukan bahwa aktivitas yang tidak optimal ini menghabiskan biaya tenaga kerja industri konstruksi lebih dari Rp. 177 Triliun.

Perangkat lunak untuk manajemen lapangan sangat penting bagi pertumbuhan industri konstruksi yang berkelanjutan. Lebih dari bidang teknologi konstruksi lainnya, perangkat lunak lapangan telah mengalami peningkatan pesat dalam inovasi dan pilihan produk bagi operator lapangan, dan hal itu juga menyebabkan “kelelahan aplikasi.” Dalam bab ini, kami akan mengulas apa yang ada di luar sana dan mengambil beberapa pelajaran untuk berpikir tentang perangkat lunak lapangan, karena perangkat lunak lapangan akan tetap menjadi salah satu bidang terpenting dalam pengembangan teknologi konstruksi.

4.1 AWAL MULA

Merupakan keluhan umum di antara tim lapangan dan teknolog konstruksi bahwa perangkat lunak lapangan awal tidak terlalu membantu. Sebagian besar perangkat lunak tersebut awalnya adalah perangkat lunak akuntansi yang diperluas ke lapangan untuk menangkap data akuntansi. Dan sebagian besar perangkat lunak tersebut memiliki pengalaman pengguna yang sama kakunya dengan perangkat lunak akuntansi, tidak terintegrasi dengan baik dengan perangkat lunak lain, dan secara umum meninggalkan kesan yang buruk. Yang terpenting, perangkat lunak awal memerlukan perubahan dalam proses harian yang tampaknya tidak dirancang untuk lapangan, tetapi bermanfaat bagi tim kantor.

Gagasan bahwa perangkat lunak dapat memaksakan proses baru pada tim yang sudah mapan merupakan gagasan lama. Pada tahun 1990-an, sektor manufaktur mengalami revolusi dengan diperkenalkannya perangkat lunak “Enterprise Resource Planning” (ERP), yang dipelopori oleh perusahaan Jerman bernama SAP. Produk ERP dari perusahaan seperti SAP, Oracle, dan lainnya sebenarnya adalah perangkat lunak kelas dunia yang benar-benar melakukan hal-hal yang bermanfaat; bahkan, produk-produk tersebut merupakan inti dari transformasi digital bagi banyak perusahaan. Namun, penerapannya jauh lebih disruptif daripada yang diperlukan saat ini, karena beberapa alasan.

ERP tradisional seperti semua perangkat lunak pada tahun 90-an – sesuatu yang Anda instal secara lokal di komputer atau server perusahaan Anda. Akibatnya, tidak ada cara bagi SAP untuk menyediakan perbaikan berkelanjutan; pada kenyataannya, setiap rilis merupakan masalah besar dan memerlukan banyak konsultan untuk menerapkannya. Jadi, implementasi ERP harus tidak fleksibel, cocok untuk semua, dan sebagai hasilnya mengharuskan pengguna untuk menyesuaikan diri dengan perangkat lunak, bukan sebaliknya. Seperti yang telah kita bahas di Bab 2, Perangkat Lunak sebagai Layanan mengubah paradigma ini sehingga

perangkat lunak kini jauh lebih fleksibel, terus berubah, dan dapat dikonfigurasi sesuai kebutuhan pengguna. Dan dengan fleksibilitas tersebut muncul perubahan yang lebih penting – perusahaan perangkat lunak kini bersaing dalam hal pengalaman pengguna, dalam membuat perangkat lunak mudah digunakan.

Saat ini, sistem ERP masih ada, dan sering dijual dengan basis SaaS, dan tidak lagi berharap dapat menuntut perubahan dari penggunanya.

4.2 MASALAH DENGAN PERANGKAT LUNAK LAPANGAN

Hampir semua orang menolak perubahan yang tidak mereka anggap perlu. Dan semua orang menolak perubahan yang mempersulit hidup mereka tanpa alasan, dan sayangnya bagi banyak orang di lapangan, hal itu terjadi karena pengalaman mereka dengan teknologi secara umum, dan perangkat lunak secara khusus. Mudah untuk menunjuk perangkat lunak akuntansi sebelumnya sebagai alasan untuk ini, tetapi ada lebih dari itu. Kenyataannya adalah bahwa sebagian besar perangkat lunak tidak dirancang untuk digunakan di lapangan, karena tidak dikembangkan oleh orang-orang di lapangan.

Proses pengembangan perangkat lunak dimulai karena seseorang memiliki kebutuhan, akan membayarnya, dan hadir ketika perangkat lunak tersebut dikembangkan untuk memastikannya berfungsi bagi mereka. Dilihat dari sudut pandang itu, apakah mengherankan bahwa perangkat lunak dimulai dengan orang-orang yang duduk di kantor, menggunakan perangkat lunak lain sepanjang hari? Jadi, perangkat lunak desain memiliki keunggulan besar dibandingkan dengan apa pun yang terkait dengan lapangan. Dan perangkat lunak akuntansi adalah yang pertama kali diperkenalkan ke lapangan, karena alasan yang sama. Orang-orang yang merancang dan membangun perangkat lunak akan selalu bias terhadap kebiasaan mereka sendiri di tempat mereka bekerja. Namun, tugas vendor perangkat lunak adalah mencari tahu di mana produk mereka akan digunakan, dan bertanya apakah mereka benar-benar membantu kru di lapangan, atau hanya membantu manajemen. Tidak ada alasan keduanya tidak mungkin benar, dan ketidakseimbangan antara tim kantor yang mudah diakses dan kru lapangan yang penting bagi keberhasilan proyek mulai ditangani. Suara lapangan mulai didengar, didorong oleh inovator cerdas, serikat pekerja, dan tim lapangan yang bersedia meluangkan waktu untuk mengajari teknolog apa yang benar-benar akan berhasil di lokasi kerja.

Jake Olsen, CEO DADO, sebuah perusahaan perangkat lunak konstruksi, menggambarkan hal ini saat ia menjelaskan bagaimana perusahaannya berdiri, dan bagaimana mereka masih mengembangkan fitur.

DADO dimulai sebagai bagian dari program inkubator teknologi yang didanai oleh DEWALT, yang berpusat di Silicon Valley. Jake dan tim awalnya datang dengan ide terkait BIM yang mereka yakini 100% akan berhasil, dan mereka mempresentasikan ide ini kepada mentor mereka di inkubator. Dalam tiga pertanyaan, mereka kembali ke papan gambar:

“Untuk siapa ini?”

“Apakah mereka membutuhkannya?”

“Apa yang sebenarnya mereka butuhkan?”

Tim DADO menghabiskan lebih dari tiga bulan untuk mewawancarai kru lapangan, dari pekerja magang hingga mandor; selain hampir setiap bagian lain dari rantai nilai konstruksi. Yang mereka temukan adalah bahwa kendala terbesar adalah menemukan hal yang tepat pada waktu yang tepat, dengan cepat dan mudah. Sasaran yang mereka dapatkan adalah untuk menghilangkan hambatan dalam mengakses informasi yang baru didigitalkan, dan pada waktunya mereka memutuskan untuk menggunakan suara sebagai cara utama untuk melakukan ini, selain antarmuka berbasis web dan aplikasi. Percakapan dengan lapangan tidak berhenti di situ – faktanya, DADO secara rutin bertemu dengan para profesional perdagangan untuk bersenang-senang, makan siang, dan forum lain yang memungkinkan mereka menguji ide-ide baru dan memastikan bahwa mereka menciptakan sesuatu yang sesuai untuk lapangan.

Jake menunjukkan bahwa banyak perangkat lunak yang tampaknya meningkatkan produktivitas hanya melakukannya untuk satu sisi persamaan informasi, biasanya sisi kantor. Alasannya, sekali lagi, adalah pekerja kantor mudah ditemukan, sudah menggunakan perangkat lunak sepanjang hari, dan segera melihat nilai dalam menggunakannya. Namun, tidak jelas bahwa produktivitas kru lapangan ditingkatkan dengan menghentikan apa yang sedang mereka lakukan, melepas sarung tangan, dan mengetik laporan harian mereka ke dalam aplikasi, dibandingkan hanya menuliskannya di selembar kertas. Manfaat bagi perusahaan secara keseluruhan sudah jelas, dan tentu saja ada baiknya mengonversi informasi semacam itu secara digital – tetapi apakah mekanisme input dioptimalkan untuk pekerja lapangan?

DADO dan yang lainnya telah melihat masalah tersebut dari perspektif lain, dengan menanyakan bagaimana pengalaman pengguna di lapangan dapat dioptimalkan untuk memastikan bahwa pengumpulan data tidak hanya mendorong kompleksitas dan inefisiensi ke lapangan. Kenyataannya adalah bahwa sebagian besar perusahaan perangkat lunak yang lebih kecil hampir tidak dapat mengeluarkan produk, sehingga mereka jarang meluangkan waktu untuk belajar dari pengguna yang kurang dapat diakses, seperti para pekerja. Ironisnya adalah bahwa banyak perusahaan rintisan begitu sibuk membangun apa yang mereka pikir dibutuhkan, mereka tidak punya waktu untuk bertanya apa yang sebenarnya dibutuhkan. Kami melihat situasi berubah, dimulai dengan perusahaan perangkat lunak manajemen konstruksi besar yang telah mulai benar-benar melayani dan mendekati para pekerja khusus. Pada saat yang sama, serikat pekerja dan asosiasi yang mewakili berbagai bidang pekerjaan, seperti United Association (UA), National Electrical Contractors Association (NECA), Mechanical Contractors Association of America (MCAA), dan Association of Union Constructors (TAUC) dan lain-lain, telah menginvestasikan banyak waktu dan uang untuk mendidik anggota mereka tentang berbagai produk perangkat lunak, dan memberikan masukan bagi perusahaan perangkat lunak untuk mulai menjembatani kesenjangan tersebut.

Namun, hal utama yang dapat diambil oleh para pembaca di bidang pekerjaan adalah bahwa penyedia teknologi pada umumnya ingin membuat hal-hal yang akan Anda gunakan, dan masukan Anda akan jauh lebih berharga daripada yang Anda kira. Jika Anda ingin mempelajari lebih lanjut, hubungi saja bluecollarlabs.com. Mereka bermitra dengan banyak

profesional konstruksi dan perusahaan rintisan perangkat lunak yang bekerja untuk menciptakan produk bagi mereka, dan akan senang mendengar dari Anda.

4.3 TINJAUAN UMUM PERANGKAT LUNAK LAPANGAN

Untuk memahami perangkat lunak di lapangan, mari kita lihat apa saja yang ada di luar sana, dari yang besar hingga yang kecil, dan dalam prosesnya pahami bagaimana platform yang baru muncul kemungkinan akan mempermudah penerapan berbagai produk perangkat lunak. Kami mulai dengan platform perangkat lunak besar yang menjanjikan untuk menyatukan seluruh proses konstruksi di bawah satu atap – perangkat lunak manajemen konstruksi.

Perangkat Lunak Manajemen Konstruksi (CM)

Ada sejumlah perusahaan yang menyediakan perangkat lunak yang akan mengelola operasi konstruksi hingga tingkat tertentu. Trimble, Autodesk, Oracle, dan Procore semuanya menawarkan platform yang akan melakukan segalanya mulai dari laporan harian hingga penjadwalan hingga manajemen BIM dan banyak lagi.

Perusahaan perangkat lunak CM berada di tengah persaingan ketat untuk mendapatkan pangsa pasar, dan semuanya terus-menerus memperkenalkan fitur baru dan fungsi yang sama sekali baru, jadi kami akan menyerahkan deskripsi produk ke situs web mereka, dan sebaliknya berfokus pada mengapa Anda harus peduli dengan perangkat lunak CM, apa yang diharapkan darinya, dan bagaimana membuatnya bekerja untuk operasi Anda.

Jadi, apa yang dicapai perangkat lunak CM? Selain mengubah kertas menjadi layar, dan membuatnya lebih mudah untuk menemukan sesuatu, apa gunanya? Untuk memahami untuk apa perangkat lunak CM ada, apa yang ingin dicapainya, dan apa yang perlu Anda ketahui tentang bagaimana perangkat lunak itu akan berubah di masa mendatang, mari kita mundur sejenak dan mempertimbangkan apa yang terjadi di lapangan. Setiap tahun, lebih dari lima juta pria dan wanita dari lebih dari 26 bidang pekerjaan yang diakui, dan sedikitnya 730.000 kontraktor dari berbagai ukuran dan spesialisasi, berkumpul untuk membuat ribuan bangunan. Setiap pekerjaan bersifat unik, setiap lokasi kerja merupakan tarian kerja yang diatur dengan cermat, atau kacau, yang dilakukan seefisien dan seaman mungkin.

Danielle Dy Buncio, Pendiri dan CEO ViaTechnik, konsultan teknologi konstruksi dan BIM, menyimpulkannya: “Konstruksi memiliki beberapa tantangan yang sama seperti proses produksi lainnya, tetapi kami juga memiliki beberapa tantangan yang benar-benar unik. Kontraktor Umum harus menyatukan, pada waktu yang tepat, banyak spesialis yang mungkin mengelola rantai pasokan mereka sendiri, dan yang mungkin tidak berada di bawah kendali kontraktual GC. Sementara itu, faktor-faktor dasar seperti cuaca dapat menghentikan semuanya secara tiba-tiba.” Kompleksitas yang ditimbulkannya berpotensi tidak terbatas, dan dalam praktiknya berarti bahwa mengelola risiko potensial ini merupakan tugas setiap orang di lokasi kerja, mulai dari tukang pipa yang memastikan ia memiliki sarung tangan dan perlengkapan pengelasan tambahan, hingga pengawas yang setiap hari mulai memeriksa lokasi untuk memastikan pekerjaan telah dilakukan, dan memastikan tidak ada kejutan.

Konstruksi telah digambarkan sebagai permainan yang mengalihkan risiko kepada pihak lain, sebuah deskripsi yang didukung oleh banyak kontrak konstruksi. Ketidakpastian konstruksi menjadikan pengalihan risiko ini sebagai dorongan alami, karena tidak peduli seberapa besar risiko yang Anda kontrakkan, akan selalu ada lebih banyak lagi.

Yang mendasari manajemen risiko ini adalah istilah kunci: ketidakpastian. Tidak dapat mengandalkan sesuatu itu mahal. Ketidakpastian tentang produktivitas berarti Anda perlu mempekerjakan pekerja tambahan. Ketidakpastian tentang cuaca berarti Anda perlu menjadwalkan hari tambahan. Ketidakpastian tentang siapa yang melakukan apa berarti Anda membuang jam kerja dalam rapat tambahan. Saya kembali ke studi PMI tentang Rp. 177 Triliun dalam jam terbuang untuk mencari informasi – itu adalah ketidakpastian murni, yang paling mahal.

Faktanya, ketidakpastian adalah cara perusahaan asuransi dan Wall Street menentukan harga risiko, karena sering kali dapat diukur. Industri keuangan mengukur ketidakpastian dengan seberapa banyak variabilitas yang ada dalam suatu proses, dan kita tahu dari studi lain bahwa produktivitas kru rata-rata dapat bervariasi hingga 100% dari hari ke hari, dan kru yang berbeda melakukan pekerjaan serupa dapat bervariasi hingga 500%. Konstruksi memiliki banyak ketidakpastian, dan karenanya memiliki banyak risiko. Ketidakpastian adalah hal yang menurunkan produktivitas, hal itu membuang-buang waktu setiap orang mulai dari CFO hingga kontraktor perencana karena mereka tidak tahu pasti di mana pekerjaan itu berada, atau apakah dua minggu ke depan akan sesuai dengan rencana, dan seterusnya.

Tugas CM Software #1: Mengurangi Ketidakpastian

Hal pertama yang perlu dilakukan perangkat lunak CM adalah mengurangi ketidakpastian. Tidak seperti solusi lainnya, perangkat lunak CM modern mampu mengurangi berbagai jenis ketidakpastian di semua tingkatan organisasi. Mari kita lihat tiga komponen ketidakpastian untuk pembuat keputusan tertentu, dan bagaimana perangkat lunak CM menangani masing-masing: informasi yang baik, analisis yang baik, dan presentasi yang baik.

Informasi yang Baik

Lokasi konstruksi adalah pusat kegiatan, dan setiap kegiatan tersebut akan mengubah beberapa aspek proyek, sehingga informasi tentang proyek terus berubah. Gabungkan hal itu dengan dampak cuaca, pemerintah, dan peraturan, serta perubahan yang berasal dari sisi desain dan pemilik, dan menjadi jelas bahwa sumber ketidakpastian yang utama adalah selalu memperbarui informasi. Di masa lalu, kelompok yang berbeda akan mendapatkan informasi pada waktu yang berbeda, sehingga wajar jika ada kekhawatiran tentang versi "kebenaran" siapa yang benar. Dampak ketidakpastian semacam ini terhadap produktivitas bisa signifikan. Produk seperti Procore, PlanGrid, Viewpoint, dan perangkat lunak CM lainnya berusaha keras untuk menghubungkan sebanyak mungkin informasi kepada pengguna dengan berbagai cara. Meskipun hal ini tidak menjamin kualitas informasi yang mendasarinya, ini berarti bahwa setiap orang di lokasi kerja dapat memiliki akses ke gambar, catatan, dan jadwal terbaru kapan saja. Istilah "satu sumber kebenaran" sering digunakan dalam konstruksi, dan itu merupakan peran utama perangkat lunak CM, khususnya di lapangan.

Jadwal, gambar, dan laporan semuanya dikumpulkan dalam satu tempat. Mengunggah gambar baru dan melacak versi dapat dilakukan dengan cepat dan otomatis, karena semuanya disimpan di satu tempat di cloud. Hal ini tentu berlaku untuk semua penyedia perangkat lunak CM utama – Procore terkenal karena kemudahan mereka dalam memberikan pelatihan pada perangkat lunak mereka secara gratis.

Jadi, bagaimana kita melakukannya – bagaimana kita memasukkan semua informasi itu ke dalam sistem perangkat lunak manajemen konstruksi? Inilah yang kita maksud ketika kita berbicara tentang transformasi digital.

"Fase" pertama dari transformasi digital konstruksi adalah mengubah semua alur kerja kertas, dan sebanyak mungkin percakapan ad hoc, menjadi bentuk digital yang memungkinkan kolaborasi instan dan pembaruan waktu nyata bagi semua orang yang membutuhkannya. Tahap pertama ini masih jauh dari kata selesai, tetapi teknologi utamanya sudah ada dan telah dikembangkan hingga siap diadopsi oleh semua tingkat perusahaan dan pekerja.

Apa yang memperlambat adopsi? Mengapa tidak semua orang menggunakan perangkat lunak CM? Perusahaan dan individu mengadopsi teknologi baru karena nilai yang diberikannya lebih besar daripada biayanya. Namun, potensi biaya yang dipertimbangkan oleh para pengadopsi bukan hanya biaya harian untuk menggunakannya, melainkan biaya mental untuk mengubah proses dan pelatihan pada perangkat lunak baru. Hal ini sering kali sesederhana tim dan pekerja yang tidak menginginkan hal lain untuk dipikirkan, padahal perangkat lunak atau kertas yang telah mereka gunakan selama bertahun-tahun tidak "rusak".

Pertanyaan tentang biaya versus nilai tingkat perusahaan juga telah dijawab dengan selusin cara, mulai dari menghitung waktu yang dihemat pengawas untuk menemukan dokumen yang tepat, hingga pengurangan kesalahan dan pengerjaan ulang, hingga RFI yang lebih cepat, dan seterusnya. Hampir semua orang yang telah mengadopsi perangkat lunak CM menyadari bahwa ini merupakan peningkatan yang sangat besar – hal ini sepadan.

Masalah yang tersisa sekarang adalah biaya yang dirasakan, rintangan mental "apakah kita benar-benar perlu melakukan ini sekarang?" Dan semakin banyak jawabannya adalah ya. Informasi yang baik bukan hanya tentang mengelola informasi, tetapi juga tentang menghasilkan data sehingga perangkat lunak CM dapat melakukan tugasnya. Konstruksi menghasilkan data dalam jumlah besar, sebagian besar dalam bentuk yang tidak berguna sebagaimana mestinya. Tahap pertama transformasi digital konstruksi ini melibatkan perusahaan yang menyadari bahwa mereka harus membuat standar internal tentang cara memasukkan informasi ke dalam perangkat lunak CM mereka – masalah yang sebelumnya tidak ada tetapi sangat penting untuk mendapatkan nilai dari perangkat lunak CM.

Misalnya, cara RFI dicatat dan diproses dapat bervariasi menurut pengawas, proyek, dan dari waktu ke waktu. Semua orang tahu bahwa proses RFI adalah salah satu masalah terbesar dalam konstruksi, dan setiap upaya untuk memperbaiki masalah keseluruhan ini harus dapat melihat data secara agregat. Akibatnya, perusahaan di seluruh industri konstruksi menstandardisasi cara memasukkan RFI, dan dengan demikian memberdayakan tim mereka untuk melacak dan menganalisis masalah dengan lebih efektif.

Lalu ada masalah menghubungkan sistem, seperti yang kita lihat di Bab 3. Banyak perusahaan, terutama kontraktor sub-perdagangan, melaporkan perlunya memasukkan informasi dua kali, terutama laporan harian, karena sistem yang mereka pilih tidak berfungsi dengan klien hulu mereka, biasanya kontraktor umum. Masalah ini sepertinya tidak akan hilang begitu saja. Banyak sistem bisnis besar tidak terhubung satu sama lain secara langsung, sehingga memerlukan pengembangan perangkat lunak penghubung yang mahal untuk menerjemahkan atau menghubungkannya.

Memang, banyak perusahaan di industri konstruksi telah menyewa konsultan seperti Jonathan Marsh, dari Steel Toe Consulting, untuk membantu mengintegrasikan berbagai sistem dan secara umum merencanakan dan mengelola transformasi digital mereka.

Secara keseluruhan, ada tiga cara informasi dari lapangan saling terhubung:

1. Semuanya berasal dari satu sistem, seperti Procore atau Autodesk Construction Cloud
2. Perusahaan membangun solusi mereka sendiri, sering kali menggabungkan produk perangkat lunak yang lebih kecil
3. Standar dikembangkan yang memungkinkan semua perangkat lunak saling menyampaikan informasi dengan lancar

Tidak satu pun dari pendekatan ini yang berfungsi sepenuhnya saat ini, sebagian besar perusahaan memadukan pendekatan pertama dan kedua, dan mengandalkan pendekatan ketiga untuk jenis file seperti file Revit atau OBJ. Sejarah koneksi data di industri lain menunjukkan bahwa hal ini dapat berlanjut untuk waktu yang lama, kecuali jika pelanggan besar atau sekelompok pelanggan menuntut pendekatan standar. Hal ini tidak mungkin terjadi, karena tidak ada satu perusahaan pun yang memiliki dominasi pasar untuk memenuhi permintaan tersebut. Ini tidak berarti inisiatif ini akan gagal, hanya saja beberapa sistem dan beberapa jenis data akan lebih mudah dihubungkan daripada yang lain. Dalam industri dengan 730.000 perusahaan, pasar tersegmentasi semacam ini hampir tidak dapat dihindari.

Yang mungkin paling mungkin adalah munculnya "jaringan API," tempat perusahaan yang lebih kecil terhubung ke platform perusahaan yang lebih besar, seperti Pro-core atau Autodesk. Itu sedang terjadi sekarang dan mungkin akan membuat interkoneksi data tidak lagi menjadi masalah di masa mendatang.

Analisis yang Baik

Hanya memiliki data dari berbagai sumber di satu tempat saja sudah merupakan nilai tambah yang besar, tetapi perangkat lunak CM dilengkapi dengan rangkaian lengkap visualisasi dan alat analisis lain yang membantu menceritakan kisah dari data tersebut. Misalnya, mengetahui pekerjaan lebih cepat dari jadwal adalah hal yang bagus, tetapi mengetahui mengapa pekerjaan lebih cepat dari jadwal adalah cara kru meningkat dari waktu ke waktu, dan bagaimana organisasi mulai mendorong produktivitas yang lebih tinggi.

Kris Lengieza, direktur pengembangan bisnis, adalah pemimpin pasar Pro-core, tempat data dari pihak ketiga diintegrasikan ke dalam sistem Procore. Sebelum bergabung dengan Procore, Kris adalah kepala bagian operasional di Stiles Construction. Kris menceritakan kisah tentang bagaimana ia menggunakan perangkat lunak CM mereka untuk menganalisis RFI, profitabilitas pekerjaan, dan ukuran pekerjaan. Kris menemukan bahwa selama bekerja di

proyek-proyek ini, manajer proyek yang menerbitkan lebih banyak RFI cenderung menjalankan proyek yang lebih menguntungkan. Namun, yang penting adalah bahwa profitabilitas tersebut berasal dari dokumentasi yang lebih baik, bukan RFI itu sendiri. Dengan mendokumentasikan RFI dan perintah perubahan yang mereka buat secara cermat, perusahaan lebih mungkin dibayar untuk perintah perubahannya.

Sekarang bayangkan dua skenario. *Dalam skenario #1,* manajemen mendatangi seorang manajer proyek yang merupakan veteran konstruksi selama 20 tahun dan memberi tahu dia bahwa mereka tidak mendokumentasikan proyek mereka dengan cukup baik. Dalam skenario ini, hanya ada beberapa contoh untuk membuktikan hal tersebut, dan PM membantah bahwa itu hanyalah pengecualian, bahwa sebagian besar waktu itu bukan masalah.

Dalam skenario #2, manajemen mengumpulkan semua PM dan menyajikan analisis komprehensif dari 50 pekerjaan, yang menunjukkan bahwa dokumentasi yang lebih tinggi menghasilkan persentase perintah perubahan berbayar yang lebih tinggi, dan hal ini menghasilkan margin yang lebih baik untuk pekerjaan-pekerjaan dengan dokumentasi yang lebih tinggi. Sekarang bayangkan bahwa manajemen telah menghitung bonus rata-rata yang akan diperoleh dari margin ini untuk seorang PM.

Perangkat lunak manajemen konstruksi seharusnya menjadi pembeda antara skenario-skenario ini, dan banyak skenario lainnya yang serupa. Peringatan ketika menyangkut analisis multisumber semacam ini. Hanya karena suatu aplikasi telah menghasilkan angka, atau keluaran, tidak berarti itu valid. Selalu, selalu terapkan insting dan skeptisisme alami Anda terhadap informasi yang berasal dari perangkat lunak. Sebagian besar waktu, informasi ini berkualitas tinggi, dan akan secara drastis meningkatkan kemampuan Anda untuk memahami dan mengelola proyek. Akan tetapi, tidak ada jaminan bahwa semua data yang dimasukkan telah dilakukan dengan benar, atau lengkap, atau ada cukup data untuk menarik kesimpulan. Kita sebagai manusia menjadi sedikit malas ketika melihat informasi dalam bentuk cetak atau digital, kita menganggapnya valid. Sebaiknya periksa ulang.

Sebagai contoh cepat, setiap kali Anda melihat persentase dalam grafik atau laporan, ada baiknya untuk menanyakan berapa banyak titik data yang diwakili oleh persentase tersebut. Grafik terkenal karena mengelabui orang dengan skala yang berbeda, atau menyajikan diagram pai berdasarkan sejumlah titik data yang sangat kecil.

Misalnya, jika Anda membaca bahwa 17,5% kru datang terlambat, ada baiknya menanyakan berapa banyak kru yang mereka survei atau kumpulkan datanya. Jika jawabannya adalah total 40 kru, itu berarti mereka mengatakan 7 kru terlambat. Itu tidak bagus, tetapi apakah 0,5% itu bermakna? Bahkan, apakah perbedaan antara 17%, 17,5%, atau 20% itu bermakna? Masing-masing persentase ini, jika diambil sebagai persentase dari hanya 40, masih sekitar 7 (40 × 0,17 = 6,8; 40 × 0,175 = 7; 40 × 0,18 = 7,2).

Ini sering disebut sebagai perbedaan antara ketepatan dan keakuratan. Dan jika analisis lebih tepat daripada akurat, itu bisa menyesatkan. Anda akan sering menemukan hal ini, dan itulah sebabnya Anda harus melihat grafik data setidaknya sebanyak Anda melihat angka – Anda akan menemukan keanehan lebih cepat, dan grafik dapat membantu Anda

melewati desimal yang terkadang mengganggu untuk melihat gambaran sederhana. Namun, selalu tanyakan berapa banyak titik data yang dianalisis untuk mendapatkan analisis tertentu.

Presentasi yang Baik

Memiliki informasi dan analisis yang baik itu penting, tetapi kemampuan untuk mengakses informasi tersebut juga penting. Karena sebagian besar perangkat lunak dapat diakses di ponsel dan laptop pengguna, Anda harus dapat melakukan ini.

Yang benar-benar membuat perbedaan adalah pengalaman pengguna dan antarmuka pengguna. Dapatkah pengguna melakukan apa pun yang mereka inginkan, kapan pun mereka membutuhkannya? Terlepas dari tingkat izin yang berbeda, di mana subkontraktor mungkin tidak dapat melihat penjadwalan atau informasi lainnya, pengguna tidak boleh merasa tidak tahu cara melakukan apa yang perlu mereka lakukan selanjutnya.

Aturan praktis sederhana untuk perangkat lunak lapangan: perangkat lunak tersebut harus terasa tidak terlihat. Seharusnya sangat mudah dan intuitif untuk digunakan sehingga pengguna lupa bahwa mereka menggunakan salah satu atau produk lainnya, dan langsung mengerjakan pekerjaan mereka dengan cepat dan mudah.

Hal tentang "intuitif" pada perangkat lunak adalah bahwa tidak ada yang secara alami intuitif tentang mengetuk layar kaca dan melihat model yang dibuat komputer di lokasi kerja yang berdebu. Intuitif benar-benar berarti ia bekerja seperti perangkat lunak yang pernah kita lihat sebelumnya, yang menggunakan pilihan desain situs web dan perangkat lunak komputer yang umum. Sebagian besar produk perangkat lunak CM melakukannya; beberapa lebih baik daripada yang lain.

Hal lain tentang menjadi intuitif adalah bahwa di setiap titik ada menu mudah yang mengarahkan Anda ke hal berikutnya yang mungkin ingin Anda lakukan. Dan di sini beberapa produk perangkat lunak jelas lebih baik daripada yang lain.

UX dan UI akan terus berubah, dan itu adalah bagian dari alasan mengapa pengujian perangkat lunak dengan personel lapangan yang sebenarnya selalu penting. UX juga sering kali dapat disesuaikan agar sesuai dengan alur kerja spesifik pengguna. Mengonfigurasi perangkat lunak agar sesuai dengan apa yang Anda butuhkan adalah hal yang umum, dan proses yang harus dianggarkan jika perangkat lunak CM, atau perangkat lunak apa pun, akan diluncurkan dalam skala besar. Seperti yang telah kita bahas di Bab 2, sebagai individu, ada baiknya Anda mempelajari cara perangkat lunak Anda menyajikan data dan opsi – jika Anda menguasai pengalaman pengguna, perangkat lunak dapat menjadi alat yang sangat berharga.

4.4 PERANGKAT LUNAK CM SEBAGAI PLATFORM

Perangkat lunak manajemen konstruksi telah mulai menjadi "sistem operasi" untuk lapangan, dan akan terus demikian di tahun-tahun mendatang. Salah satu cara melakukannya adalah dengan menerima data dan fungsi dari aplikasi lain melalui API mereka. Semua platform CM besar memiliki program khusus yang memudahkan pemilihan dan pembelian aplikasi tambahan ini, dengan integrasi hanya dengan satu atau dua klik. Procore memiliki pasar Aplikasi mereka sendiri, Autodesk memiliki Autodesk Construction Connect, Trimble dan

CMiC keduanya memiliki program integrasi khusus yang membantu penyedia melakukan integrasi, dan pengguna dapat menghubungkan aplikasi ini ke layanan inti mereka.

Inovasi yang dapat dilepaskan oleh platform seperti ini sungguh luar biasa, karena seluruh ekosistem startup, dengan ratusan pengembang, perusahaan kecil, dan desainer produk yang inovatif kini dapat menambahkan satu pengalaman pengguna. Salesforce telah membuktikan model ini di bidang penjualan dan pemasaran, dan saya yakin kita akan melihat hal yang sama dalam konstruksi.

Tugas #2 CM Software: Meningkatkan Produktivitas

Risiko dan ketidakpastian memang penting, tetapi produktivitas juga penting. Perlu dijelaskan apa yang kami maksud ketika kami mengatakan "produktivitas," karena kebanyakan orang tidak menghabiskan banyak waktu untuk memikirkan aktivitas mereka dalam hal nilai per jam, tetapi itulah intinya. Produktivitas diukur dalam bentuk nilai dolar yang diproduksi dalam satu jam, dirata-ratakan di seluruh kru, perusahaan, dan industri. Sebagai manajer perorangan, Anda mungkin menghitung unit pekerjaan yang diselesaikan, tetapi untuk menganalisis kelompok pekerja yang besar, kami menggunakan nilai dolar.

Data Biro Statistik Tenaga Kerja telah banyak dikutip sebagai gambaran pertumbuhan produktivitas konstruksi yang rendah. Mari kita bahas hal tersebut, sehingga kita dapat memahami apa yang sebenarnya dapat dilakukan oleh pekerja garis depan dan manajer terkait produktivitas, dan bagaimana perangkat lunak membantu.

Ada dua jenis aktivitas pada setiap proyek. Aktivitas yang menambah nilai moneter, dan aktivitas yang tidak. Karena produktivitas diukur dengan dolar, bukan sumber nilai lainnya (seperti keselamatan atau ketepatan waktu), segala sesuatu yang dilakukan oleh perusahaan yang tidak secara langsung menghasilkan sesuatu yang dapat dibayar bukanlah nilai tambah yang digunakan untuk perhitungan produktivitas.

Untuk memahami produktivitas, kita dapat berasumsi bahwa ada tiga jenis aktivitas yang terjadi dalam konstruksi: Kategori pertama adalah aktivitas yang secara langsung menghasilkan pembayaran, yang dalam kasus konstruksi hanya berupa pekerjaan. Kategori kedua adalah aktivitas yang secara langsung mendukung pekerjaan, seperti inspeksi keselamatan, perencanaan, dan sebagainya. Kategori ketiga terdiri dari banyak hal yang menyita waktu tetapi tidak memajukan proyek, seperti pengerjaan ulang, dan biaya koordinasi seperti menunggu satu kru selesai sehingga kru berikutnya dapat memulai. Sasaran dari setiap proyek adalah untuk menghabiskan persentase waktu terbesar yang memungkinkan untuk langsung mengerjakan sesuatu, menghemat waktu dan biaya dari aktivitas pendukung, dan menghilangkan biaya pengerjaan ulang dan koordinasi. Perangkat lunak CM hadir untuk mewujudkan semua hal ini. Mari kita tinjau dua tugas utama untuk mengurangi dukungan dan menghilangkan pemborosan, melalui sudut pandang bagaimana Anda, operator di lapangan, dapat memanfaatkan apa yang ditawarkan perangkat lunak:

1. Menghemat waktu dan biaya dari aktivitas pendukung: Proyek yang kompleks memerlukan waktu yang signifikan hanya untuk melacak apa yang sedang terjadi. Perangkat lunak CM membantu dalam empat cara:

a. *Mengelola perencanaan:* kini ada satu tempat, yang diperbarui secara otomatis dengan segera, tempat rencana berada. Sistem ini mencatat, secara waktu nyata, pekerjaan yang dilakukan untuk rencana tersebut, kejadian yang mengganggu rencana (seperti RFI), perubahan rencana dari tim desain, dan sebagainya.

   Apa yang harus Anda ketahui dan lakukan: Sistem ini biasanya tidak mengharuskan tim mengubah apa pun tentang proses mereka, kecuali bahwa mereka perlu memasukkan informasi secara konsisten. Pastikan untuk menstandardisasi istilah yang digunakan, klasifikasi, dan hal lain yang dapat distandardisasi di seluruh tim, sehingga manajer dan analis dapat memanfaatkan sistem untuk laporan tingkat yang lebih tinggi.

b. *Mengelola gambar:* Setiap orang pada akhirnya bertanggung jawab untuk menyampaikan gambar yang disetujui. Menemukan gambar yang tepat dan memastikannya benar-benar gambar terkini, merupakan proses yang mahal di masa lalu. Perangkat lunak modern memastikan bahwa gambar terkini tersedia setiap saat, di laptop, tablet, atau ponsel. Apa yang harus Anda ketahui dan lakukan: Luangkan waktu untuk memahami dengan tepat cara mengoperasikan perangkat lunak – ini dirancang agar mudah, tetapi akan sepadan dengan usaha untuk mendapatkan sedikit pelatihan melalui tutorial daring atau sekadar bertanya kepada rekan kerja.

c. *Mengelola pekerjaan aktual:* Laporan harian adalah satu-satunya sumber informasi berkelanjutan terpenting bagi manajer dan tim proyek, karena laporan tersebut memungkinkan pekerjaan yang biasa-biasa saja dan berkelanjutan dipahami tanpa penyaringan memori dan kejadian terkini. Tidak seorang pun mengingat kejadian kecil tiga bulan lalu, tetapi dengan membuat laporan harian yang baik, kejadian tersebut dapat dengan mudah diingat kembali saat dibutuhkan. Dan tentu saja, jika terjadi perselisihan, dokumentasi terbaiklah yang menang.

   Lebih luas lagi, seiring berjalannya waktu kita akan menemukan bahwa informasi dari laporan harian, serta masukan data rutin lainnya, akan digunakan untuk membuat rekomendasi perbaikan proses baru.

   Apa yang harus Anda ketahui dan lakukan: Dorong kru Anda untuk memasukkan informasi sekonsisten mungkin, termasuk insiden dan sumber gesekan proses.

d. *Koordinasi:* Gambar, rencana, jam, keselamatan, dan inspeksi semuanya mengharuskan banyak tim yang berbeda, beberapa di antaranya tidak terhubung secara kontrak satu sama lain, bekerja sama dan meminimalkan waktu yang terbuang. Perangkat lunak CM adalah sumber kebenaran utama yang dapat dimanfaatkan untuk meminimalkan upaya dan waktu koordinasi.

   Apa yang harus Anda ketahui dan lakukan: Pelajari cara menggunakan fitur penjadwalan dan notifikasi perangkat lunak CM, fitur tersebut dapat membuat

pelacakan tanggal inspeksi penting dan acara penting lainnya menjadi jauh lebih mudah.

2. Menghilangkan pemborosan: Banyak fungsi perangkat lunak yang sama yang akan menghasilkan dukungan kerja yang lebih efisien juga akan menghasilkan setidaknya penghapusan pemborosan sebagian. Kenyataannya, tentu saja, akan selalu ada waktu yang terbuang untuk aktivitas yang tidak produktif, tetapi jika kita dapat memanfaatkan kemudahan pencarian dokumen, satu sumber gambar yang paling mutakhir, dan akses langsung ke informasi yang paling mutakhir, waktu yang dihabiskan untuk mencari informasi, menunggu pengambilan dokumen, dan aktivitas pemborosan lainnya dapat diminimalkan.

Pekerjaan Perangkat Lunak CM #3: Peningkatan Proses

Digitalisasi alur kerja kertas biasanya dianggap sebagai fase pertama transformasi digital konstruksi. Hanya dengan mengubah kertas menjadi perangkat lunak akan menciptakan semua nilai yang dijelaskan sebelumnya. Namun, itu tidak benar-benar mengubah bisnis, hanya membuat dokumen lebih cepat.

Nilai sebenarnya mulai muncul dari penggunaan perangkat lunak dan data untuk menganalisis cara kerja berbagai hal dan menemukan peningkatan. Peningkatan proses terjadi karena seseorang bersedia meluangkan waktu dan upaya untuk meyakinkan semua orang yang menggunakan proses tersebut bahwa ada cara yang lebih baik.

Dalam industri tempat banyak orang telah melakukan pekerjaan mereka selama beberapa dekade, dan dapat menunjuk gedung-gedung besar untuk menunjukkan bahwa cara lama tidak rusak, bagaimana Anda membuat orang melakukan sesuatu dengan cara baru?

Jawabannya adalah membuktikan bahwa Anda benar. Satu-satunya cara untuk membuktikan bahwa proses baru lebih baik adalah melalui data. Dalam Bab 1, kita membahas perbedaan penerimaan antara teknologi yang dapat Anda lihat, dan teknologi yang tidak dapat Anda lihat. Kesenjangan yang sama ini berlaku untuk efek yang dapat Anda lihat dengan mata kepala sendiri, dan efek yang memerlukan teknologi untuk melihatnya. Mudah untuk melihat bahwa memberi semua orang akses ke informasi yang sama pada waktu yang sama akan membantu. Yang tidak begitu jelas adalah bagaimana menganalisis data dapat mengungkap beberapa asumsi sebagai tidak benar, atau menunjukkan kepada Anda bahwa tim menghabiskan lebih banyak waktu pada beberapa bagian dari proses pembangunan daripada yang Anda kira.

Menggunakan teknologi untuk melihat hal-hal yang tidak dapat Anda lihat dengan mata telanjang bukanlah hal baru. Ahli listrik mendasarkan seluruh profesi mereka pada penggunaan teknologi untuk melihat amplitudo, voltase, frekuensi, dan sebagainya. Semua orang mengandalkan laser untuk mengukur dimensi dan perataan lokasi kerja, dan Anda tidak dapat melihat keduanya.

Perangkat lunak CM, untuk pertama kalinya, memungkinkan para manajer di lapangan, serta di kantor, untuk melihat bagaimana proses berjalan pada tingkat yang tidak mungkin dilihat secara langsung. Keputusan dapat dibuat berdasarkan data, bukan pengalaman atau

intuisi. Dan begitulah cara Anda membuktikan bahwa Anda benar, bukan sekadar bersuara lantang.

Todd Mustard, VP Inovasi Industri di The Association of Union Constructors (TAUC) langsung terjun ke dunia bisbol saat kami membahas hal ini. Sekarang, semua orang tahu kisah Oakland A's dan bagaimana mereka menggunakan data tentang pemain dan tim untuk memahami hal-hal yang tidak dapat ditangani oleh naluri saja, dan dalam prosesnya mereka menciptakan tim yang menang.

Inilah yang dapat disediakan oleh perangkat lunak CM, tetapi untuk melakukannya, data harus dimasukkan secara konsisten, benar, dan kemudian dianalisis dengan baik. Bayangkan jika Anda mengambil semua tumpukan data yang dihasilkan oleh konstruksi dan membuatnya tersedia untuk dianalisis? Sejarah dan pengalaman industri lainnya mengatakan bahwa hampir semua hal tentang pekerjaan menjadi lebih baik, dari efisiensi hingga keselamatan hingga hal-hal sederhana seperti tingkat kenyamanan. Berikut adalah tiga contoh bagaimana perusahaan telah mengambil data yang disediakan oleh perangkat lunak CM, menganalisis data tersebut, dan membuat keputusan penting yang secara dramatis memengaruhi operasi mereka:

1. *Penaksir:* Di Australia, Scott Polson dari Benmax menceritakan bagaimana mereka dapat menggunakan perangkat lunak CM mereka untuk menganalisis pekerjaan apa yang menghasilkan uang, dan mana yang tidak. Di masa lalu, mereka mungkin menerima pekerjaan karena mereka mengenal pengembang atau pemilik, tetapi mereka menemukan bahwa beberapa dari hubungan tersebut sebenarnya tidak menguntungkan. Benmax sekarang dapat membanggakan tingkat kemenangan 60% pada penawaran.
2. *Menaksir (lagi):* Contoh CJ Best tentang penggunaan laporan harian untuk benar-benar memahami waktu yang dihabiskan pada proses VDC tertentu mengubah pemahaman mereka tentang biaya, dan mengarah pada tinjauan estimasi pekerjaan.
3. *RFI*: Contoh konstruksi Stiles Kris Lengieza menunjukkan nilai analisis; Hal ini juga menunjukkan bagaimana Stiles mampu mengidentifikasi arsitek/insinyur yang lambat dalam menanggapi RFI, sehingga memberikan tim Stiles kemampuan untuk berfokus dalam memperoleh jawaban dari mereka dengan lebih cepat.

Pekerjaan CM Software #4: Sumber Kebenaran Tunggal, di Mana Saja

Perangkat lunak CM memungkinkan setiap orang yang terlibat dalam suatu pekerjaan untuk melihat gambar, laporan, analisis, dan informasi relevan pekerjaan lainnya yang mungkin mereka perlukan, tergantung pada izin berbasis peran. Ini sudah penting sebelum pandemi 2020, tetapi karena pembatasan sosial telah berubah menjadi pembatasan sosial dan tindakan lain untuk mengurangi penyebaran virus, banyak asumsi tentang perlunya melakukan sesuatu secara langsung terbukti kurang akurat daripada yang diasumsikan. Kita tidak selalu perlu bersama untuk melihat revisi gambar dan mendiskusikannya.

Fakta bahwa teknologi berbasis cloud seperti perangkat lunak CM memungkinkan pembaruan waktu nyata, pemanggilan gambar dan informasi lainnya secara instan membuat peralihan ke tim virtual hampir seketika.

Tanpa menggunakan perangkat lunak semacam ini, tim akan menyimpan gambar dan informasi dalam email, atau penyimpanan cloud (misalnya, Box atau Dropbox), yang menimbulkan masalah tentang siapa yang dapat mengakses, melacak versi, dan tentu saja penundaan antara menandai sesuatu dan memasukkannya ke folder yang tepat. Dampak pada pengerjaan ulang, yang sudah menjadi manfaat besar, akan jauh lebih mendalam ketika lebih sedikit manajer dan anggota tim desain yang dapat mengunjungi lokasi secara rutin.

Saat buku ini mulai dicetak, semua penyedia perangkat lunak manajemen konstruksi telah sibuk meluncurkan fitur yang mendukung tim jarak jauh, dan seiring waktu hal ini kemungkinan akan berkembang menjadi rangkaian fitur yang mendukung pengaturan kerja hibrida apa pun yang muncul di tahun-tahun setelah pandemi dan pengunciannya. Kita dapat mengharapkan beberapa pengembalian ke alur kerja pra-pandemi tetapi juga berharap bahwa, sekarang setelah perusahaan menyadari bahwa produktivitas tidak menguap dalam tim jarak jauh, beberapa tingkat kerja jarak jauh akan tetap ada.

4.5 PERANGKAT LUNAK LAPANGAN DI LUAR CM

Tidak ada kekurangan perangkat lunak yang dibuat untuk menangani bagian-bagian kecil dari proses konstruksi – terlalu banyak jenis dan perusahaan untuk disertakan di sini. Namun, sebagian besar keputusan yang mungkin Anda buat akan tentang solusi yang lebih kecil, bukan tentang perangkat lunak CM yang lebih besar.

Tiga hal yang harus Anda pahami dan pertimbangkan tentang perangkat lunak baru yang ingin Anda adopsi: Apakah perangkat lunak tersebut berfungsi untuk semua pengguna Anda; apakah perangkat lunak tersebut berfungsi dengan perangkat lunak CM Anda; apakah Anda memiliki tim untuk mengelolanya?

Apakah Ini Berfungsi untuk Semua Pengguna Anda?

Sudah menjadi hal yang lumrah jika hanya satu bagian dari tim yang diajak berkonsultasi saat perangkat lunak lapangan baru dibeli. Seperti yang ditunjukkan oleh penelitian oleh Dodge Analytics dan lainnya, di masa lalu, perangkat lunak lapangan dibeli tanpa mengujinya dengan personel lapangan.

Seperti yang akan kita pelajari dalam bab tentang inovasi dan adopsi, banyak kontraktor umum telah membuat kelompok inovasi, dan hampir semuanya melalui proses yang sama. Pada tahun pertama, mereka membeli banyak teknologi, yang tidak digunakan oleh siapa pun. Pada tahun kedua, mereka mulai bertanya kepada kru apa yang mereka inginkan. Pada tahun ketiga, mereka semua menyiapkan proses formal yang bekerja dengan tim lapangan untuk memahami kebutuhan dan mencocokkannya dengan perangkat lunak dan teknologi lain yang telah mereka dapatkan di pasar.

Sangat mudah bagi 100 perusahaan kontraktor umum teratas untuk memiliki tim seperti ini, dan banyak yang melakukannya. Namun, dalam industri yang memiliki lebih dari 650.000 perusahaan dengan jumlah karyawan di bawah 250 orang, sebagian besar tidak mampu mempekerjakan tim untuk mengelola inovasi. Tidak memiliki kelompok inovasi seharusnya tidak menghentikan Anda untuk mendapatkan manfaat dari teknologi baru, tetapi

Anda tetap memerlukan proses untuk memastikan investasi Anda dalam teknologi baru membuahkan hasil.

Untuk perusahaan tanpa tim inovasi, proses empat langkah sederhana dapat membantu Anda memutuskan teknologi apa yang layak dicoba. Proses ini menciptakan semacam "corong", di mana setiap langkah akan mendiskualifikasi lebih banyak pilihan, hingga Anda berhasil menyingkirkan semuanya, atau menemukan beberapa pilihan. Asumsinya adalah jika Anda mencari, Anda telah mengidentifikasi pertanyaan atau masalah – sesuatu yang menurut Anda dapat dilakukan dengan lebih baik. Jadi, kita mulai dari sana.

Langkah 1: Tentukan masalahnya. Berikan tim Anda waktu 90 menit untuk benar-benar memetakan apa masalahnya. Panggil kru lapangan, dan ajukan pertanyaan seperti:

- Apa yang tidak berhasil?
- Bagaimana kita mengukur pekerjaan yang akan terpengaruh? – Siapa yang mengerjakan pekerjaan tersebut?
- Siapa yang akan mengoperasikan perangkat lunak tersebut? Apakah ada lebih dari satu kelompok?
- Apa yang mereka lakukan sekarang?
- Apakah perangkat lunak yang Anda lihat sesuai dengan yang disebutkan di atas?

Langkah 2: Dapatkan demo. Demo langsung jika memungkinkan. Yang penting di sini adalah memahami bahwa sebagian besar perusahaan perangkat lunak buruk dalam mendemonstrasikan produk mereka – mereka akan menunjukkan bagian-bagian mengilap yang mereka banggakan, tetapi sering kali tidak memperhatikan apa yang Anda pedulikan. Jadi, beri tahu mereka apa yang Anda pedulikan, dan beri tahu mereka bahwa Anda ingin tim Anda dapat mencoba produk tersebut dalam demo jika memungkinkan. Bagi banyak produk perangkat lunak, ini hanya berarti mengunduh sesuatu dan mendapatkan nama pengguna sementara.

Vendor perangkat lunak mungkin menolak ini, tetapi tidak seorang pun di tim Anda dapat diharapkan untuk benar-benar memahami perangkat lunak tersebut jika mereka tidak menyentuhnya. Anda akan menganggukkan kepala dan sebagian besar akan mengerti, tetapi tidak ada yang mengalahkan menggunakannya, meskipun hanya selama 20 menit.

Ini juga merupakan detektor BS yang bagus, karena banyak, banyak perusahaan perangkat lunak tahap awal akan memberi tahu Anda bahwa perangkat lunak tersebut dapat melakukan hal-hal yang tidak dapat dilakukannya, setidaknya belum. Ini juga akan menjadi cara yang baik untuk melihat seberapa yakin mereka bahwa perangkat lunak tersebut akan bebas bug. Ini juga merupakan cara yang fantastis untuk membuat tim Anda tetap terlibat dalam teknologi, dan merasa bahwa mereka adalah bagian dari proses pengambilan keputusan. Dalam kasus terburuk, ini adalah biaya pelatihan. Dalam kasus terbaik, Anda mendapatkan pemahaman awal tentang apa yang mungkin diperlukan untuk peluncuran.

Sebelum rapat ini, luangkan waktu 15 menit untuk memikirkan apa yang diperlukan agar mereka menang, dan sampaikan hal itu kepada mereka sehingga mereka membicarakannya dan Anda tidak membuang waktu untuk hal-hal yang tidak Anda perlukan.

Langkah 3: Dapatkan uji coba. Anda harus membayar untuk ini, mungkin sedikit lebih mahal dari yang mereka tetapkan secara resmi. Ini sepadan dengan uang yang dikeluarkan untuk memastikan Anda mendapatkan waktu dan dukungan yang Anda butuhkan. Uji coba mungkin mengharuskan perusahaan perangkat lunak melakukan beberapa konfigurasi dan pengaturan untuk Anda, dan Anda harus meminta mereka juga memberikan pelatihan untuk kru uji coba Anda.

Semoga Anda bisa mendapatkan demo langsung, sehingga Anda masuk ke uji coba dengan beberapa ide tentang apa yang berhasil dan apa yang tidak Anda yakini. Uji coba harus 100% tentang nilai yang diberikan pada proyek, dikurangi upaya yang diperlukan untuk membuat perangkat lunak berfungsi.

Vendor perangkat lunak harus memiliki meja bantuan atau dukungan mudah lainnya, dan tim penjualan yang dapat Anda temui secara rutin untuk memastikan Anda mendapatkan semua nilai dari perangkat lunak yang Anda bisa. Mereka akan menyarankan tiga bulan, tetapi Anda harus merasa nyaman meminta enam bulan jika Anda merasa itu perlu, seperti yang dilakukan banyak kontraktor.

Pastikan untuk bertanya pada diri sendiri:

- Apakah ini bekerja dengan sempurna?
- Apakah ini menghemat waktu secara keseluruhan? Apakah ini mengalihkan kesulitan dari kantor ke lapangan, atau sebaliknya?
- Apakah penghematan ini cukup untuk sepadan dengan kompleksitas dan biaya tambahan?

Langkah 4: Peluncuran. Dengan asumsi semuanya berjalan dengan baik, Anda akan ingin meluncurkannya di antara tim Anda. Sekarang beberapa orang di tim Anda mengetahui produk dengan baik, sehingga dapat menjadi pendukungnya. Namun, Anda juga harus bergantung pada perusahaan perangkat lunak untuk dukungan dalam meluncurkannya – ingatlah bahwa mereka menghasilkan lebih banyak jika Anda membeli lebih banyak.

Sebagian besar perusahaan perangkat lunak akan memiliki kelompok yang disebut "kesuksesan pelanggan," dan tugas mereka adalah bekerja dengan pelanggan untuk mendorong dan melacak adopsi internal produk. Mereka akan sering melakukan webinar, menerima panggilan tim lapangan, dan menyediakan materi yang membantu Anda saat proses berjalan. Salah satu hal utama yang terlewatkan dalam peluncuran awal adalah cara memasukkan data, atau cara lain menggunakan perangkat lunak sehingga manajemen dapat melihat bagaimana setiap orang menggunakan produk tersebut sekaligus. Konsistensi data dan konten yang dihasilkan oleh perangkat lunak sering kali menjadi salah satu nilai tambah besar yang dapat diberikan oleh perangkat lunak, jadi mintalah tim keberhasilan pelanggan untuk membantu Anda memikirkan hal ini.

Apakah Ini Berfungsi dengan Perangkat Lunak CM Anda?

Dulu, produk perangkat lunak besar seperti Autodesk, Trimble, dan CMiC hanya berfungsi dengan produk perangkat lunak saudaranya sendiri. Faktanya, hal ini berlaku di seluruh industri teknologi, karena Microsoft, IBM, Apple, dan pemain besar lainnya mengunci sistem mereka selama beberapa dekade.

Namun, munculnya API dan komputasi awan telah menyebabkan terbukanya sistem, dengan vendor perangkat lunak besar yang semakin menjadi platform. Faktanya, satu cara yang sangat mudah untuk mulai mencari perangkat lunak yang dapat memecahkan masalah tertentu yang Anda miliki adalah dengan mengunjungi perangkat lunak CM yang Anda gunakan dan meninjau mitranya. Procore secara khusus memiliki seluruh pasar dengan ratusan mitra – dalam kasus ini, perusahaan-perusahaan ini tidak hanya bekerja dengan API, mereka juga telah diperiksa oleh Procore, jadi Anda memiliki tingkat keyakinan tertentu bahwa perangkat lunak tersebut akan berfungsi seperti yang diiklankan.

Namun, banyak perusahaan lain yang bukan mitra formal tetapi memiliki API yang akan berfungsi dengan perangkat lunak CM. Ini adalah salah satu contoh di mana uji coba merupakan ide yang bagus, karena memiliki koneksi API tidak berarti mereka telah benar-benar mengetahui cara bekerja dengan baik dengan perangkat lunak CM yang Anda gunakan. Sebagai contoh, Procore memiliki lebih dari selusin "kategori" API hanya untuk laporan harian. Cara Procore memilih untuk mengatur informasi laporan harian mungkin tidak sama dengan cara yang dilakukan vendor potensial Anda, jadi Anda perlu melihat apakah mereka dapat membuatnya berfungsi. Biasanya mereka bisa, tetapi dapat menyebabkan penundaan yang nyata jika mereka harus mencari tahu selama penerapan Anda. Kesulitannya di sini adalah bahwa banyak kontraktor khusus tidak hanya memiliki satu perangkat lunak CM yang mereka gunakan, karena mereka akan menggunakan apa yang dibutuhkan oleh GC yang berbeda untuk pekerjaan yang berbeda. Kami akan membahasnya sebagai berikut.

Apakah Anda Memiliki Tim untuk Mengelolanya?

Kita baru saja sampai pada titik awal munculnya beberapa perangkat lunak yang sangat menarik yang dapat melakukan hal-hal menakjubkan, seperti penangkapan realitas, kecerdasan buatan, dan banyak lagi. Sering kali, hal-hal menakjubkan ini memerlukan sumber daya semi-khusus untuk mengelolanya.

Untuk beberapa teknologi, seperti pesawat nirawak, hal ini jelas. Namun, untuk banyak jenis perangkat lunak yang hanya mendigitalkan proses kertas, atau mengumpulkan beberapa jenis informasi, awalnya tidak jelas bahwa seseorang, biasanya sekelompok kecil, perlu menggembalakan perangkat lunak tersebut selama satu tahun atau lebih agar tim Anda terbiasa dengannya, dan mulai benar-benar mengubah proses mereka untuk memanfaatkan apa yang dapat dilakukan oleh perangkat lunak tersebut.

Tidak ada yang bekerja secara ajaib begitu saja, dan sebagian besar nilai yang akan Anda dapatkan dari perangkat lunak baru akan berasal dari perubahan tak terduga dalam cara orang bekerja, pertanyaan apa yang dapat Anda jawab, dan keputusan apa yang dapat dibuat dengan lebih cepat atau meyakinkan. Itu membutuhkan waktu dan upaya yang terfokus, waktu yang harus dialokasikan di awal dan secara sadar, atau Anda berisiko mengalami kegagalan uji coba dan penerapan yang mengikutinya. Di dunia saat ini, di mana teknologi telah menjadi bagian dari keunggulan kompetitif, jika Anda tidak memanfaatkan pilihan perangkat lunak dan teknologi secara maksimal, Anda akan dirugikan dibandingkan perusahaan lain.

Pekerjaan Khusus Dibandingkan dengan Penggunaan Perangkat Lunak Lapangan oleh Kontraktor Umum

Bagian terakhir ini berkaitan dengan kesulitan yang dihadapi oleh pekerjaan khusus dalam membuat hampir semua keputusan teknologi. Sebagai permulaan, perusahaan kontrak perdagangan sering kali lebih kecil daripada GC, sehingga jarang memiliki tim inovasi yang besar.

Namun, masalah yang lebih besar adalah bahwa kontraktor perdagangan khusus dipekerjakan oleh GC, yang mengelola lokasi, dan menentukan teknologi mana yang digunakan. Akibatnya, kontraktor perdagangan biasanya perlu mendukung beberapa produk perangkat lunak CM sekaligus, dan sangat sering menggunakan teknologi yang telah dikembangkan untuk GC, bukan untuk mereka. Hal ini berubah karena Procore, Autodesk, dan yang lainnya semakin berfokus pada pekerjaan, meskipun masih ada pekerjaan yang harus dilakukan untuk menangkap detail lokasi kerja tingkat tinggi yang dibutuhkan oleh kru pekerjaan.

Ada beberapa vendor perangkat lunak, seperti eSub, yang secara khusus menargetkan pekerjaan khusus. Dalam kasus eSub, mereka menyediakan sebagian besar fungsi yang sama dengan yang disediakan CM untuk GC, hanya saja diarahkan pada masalah unik yang dihadapi kontraktor khusus.

Apa pun perangkat lunak yang ingin Anda adopsi, pembahasan di atas sama berlaku untuk kontraktor khusus seperti halnya untuk kontraktor umum, karena sebagian besar tindakan pencegahan dan langkah yang telah saya uraikan lebih banyak tentang kesulitan yang dihadapi perusahaan mana pun yang mengadopsi perangkat lunak daripada tentang hal-hal spesifik tentang cara penggunaannya.

Secara umum, Anda tidak dapat berasumsi bahwa pengembang perangkat lunak memahami proses kerja, keterampilan, dan kondisi tempat Anda bekerja. Mengharapkan bahwa vendor perangkat lunak kecil akan memahami setiap perdagangan tidaklah realistis, tetapi Anda dapat bekerja sama dengan mereka untuk mengambil beberapa inovasi ini dan membuatnya bekerja untuk Anda – seperti hampir semua orang telah menggunakan Excel selama bertahun-tahun, meskipun faktanya tidak ada "Excel untuk Pipefitter."

Pendekatan yang baik adalah meminta mereka menemui Anda di tempat Anda berada, dari sudut pandang persyaratan dan tingkat kenyamanan. Lakukan Langkah 1–4 yang disebutkan sebelumnya, tetapi fokuslah juga pada bidang usaha spesifik Anda. Mintalah contoh kasus dan referensi untuk bidang usaha Anda, atau bidang usaha yang berdekatan.

Dalam semua perangkat lunak lapangan, seperti halnya dalam semua teknologi konstruksi, yang terpenting untuk diingat adalah bahwa semua teknologi dan produk ini ada untuk Anda. Pekerja terampil membawa pengalaman, intuisi, dan penilaian, yang membuat dunia yang dibangun berdiri kokoh. Peralatan, tidak peduli berapa banyak lampu yang menyala, hanya membantu Anda melakukannya sedikit lebih baik.

Lebih banyak yang telah ditulis tentang BIM daripada teknologi lainnya di AEC. Sebagian besar dari apa yang ditulis lebih terkait dengan arsitektur daripada konstruksi,

karena sebagian besar penggunaan BIM berasal dari sisi desain. Itu berubah, jadi mari kita lihat BIM dari perspektif konstruksi.

BIM

BIM adalah kata lain yang digunakan tanpa resolusi yang jelas, jadi mari kita mulai dari sana. BIM, tentu saja, berarti "model informasi bangunan". Itu berarti kumpulan informasi tentang suatu bangunan. Itu tidak berarti file Revit, atau file Rhino, meskipun keduanya dapat menghasilkan model BIM. BIM sering disalahartikan sebagai model geometris 3D yang hampir selalu disertakan dalam BIM modern. Namun, BIM bukanlah model 3D, melainkan model yang dimaksudkan sesuai dengan namanya – model informasi tentang sebuah bangunan.

Seperti yang disampaikan oleh Dana Smith dan Michael Tardif dalam buku fantastis mereka Building Information Modeling: "Geometri sebuah bangunan hanya mewakili sebagian kecil dari keseluruhan informasi yang berguna tentang bangunan itu. Model informasi bangunan yang benar-benar komprehensif tidak hanya mencakup geometri tetapi juga semua informasi tentang bangunan yang dibuat sepanjang masa manfaatnya." Definisi ini bermasalah, karena "semua informasi yang dibuat sepanjang masa manfaatnya" berarti segala hal mulai dari desain melalui konstruksi hingga serah terima ke manajemen fasilitas hingga pembongkaran. Tidak ada format yang disepakati untuk informasi sebanyak itu, tidak ada standar tentang bagaimana setiap orang yang membutuhkan informasi semacam itu dapat mengabaikan semua informasi yang tidak perlu mereka perlukan. Bagi kru konstruksi, "masa manfaat" berarti sejak rencana terakhir yang disetujui; riwayat perubahan desain bukanlah sesuatu yang harus dilihat oleh kru lapangan, karena sudah cukup sulit untuk membuat semua orang mengetahui rencana terbaru. Yang penting bagi konstruksi tentang BIM adalah bahwa ia telah mulai memformalkan kemampuan desain yang selalu dimiliki oleh perdagangan khusus untuk perakitan dan pekerjaan mereka sendiri yang akan dilakukan. Faktanya, yang lebih penting dari BIM itu sendiri adalah gerakan menuju desain dan konstruksi virtual (VDC) dalam perdagangan dan kontraktor umum.

BAB 5
KONSTRUKSI INDUSTRI

Terdapat ketegangan dalam industri konstruksi, antara kebutuhan untuk perbaikan dan kebutuhan untuk menjaga risiko tetap rendah. Dalam industri yang secara inheren berisiko, seperti konstruksi, refleksnya adalah selalu mempertahankan apa yang berhasil dan membangun di atasnya, bukan mengubahnya dengan metode yang kurang dipahami dan kurang intuitif.

Perangkat lunak akan mendigitalkan alur kerja saat ini, dan itu akan membuat perbedaan dalam hal keselamatan dan efisiensi pekerjaan konstruksi. Namun, untuk membuat lompatan nyata dalam keselamatan, efisiensi, dan produktivitas, dan menarik generasi pekerja baru yang memiliki pilihan lain dan lebih suka menghindari menghabiskan sepanjang hari di luar, konstruksi industri harus menjadi lebih umum. Pandemi tahun 2020 diperkirakan akan menambah tren ini, karena jauh lebih mudah untuk menegakkan jarak sosial di pabrik daripada di lokasi kerja.

5.1 JENIS KONSTRUKSI INDUSTRI

Apa yang kita maksud dengan "konstruksi industri?" Di sini satu istilah digunakan untuk mengartikan banyak hal, jadi mari kita mulai dengan mendefinisikan apa arti konstruksi industri.

Konstruksi terindustrialisasi adalah kumpulan metode yang menerapkan teknik manufaktur pada konstruksi, dan mencakup hal-hal berikut:

1. ***Prefab:*** Perakitan satu kali yang dilakukan di lokasi selain lokasi pemasangan akhir. Prefab dapat dilakukan di luar lokasi, atau di area persiapan di lokasi kerja. Secara umum, prefab adalah pekerjaan perakitan untuk satu pekerjaan, dan tujuan melakukannya secara terpisah dari lokasi akhir adalah kemampuan untuk memisahkan jadwal pembangunan perakitan tertentu dengan jadwal di lokasi, yang penting bagi kontraktor khusus.
2. ***Flat pack***: Perakitan yang dibuat dalam jumlah besar, di mana perakitan yang sama dibuat beberapa kali dan dikemas untuk diangkut, kemudian dibongkar dan dipasang di lokasi. Produksi flat pack lebih diuntungkan dari skala ekonomi yang memberikan keuntungan biaya dan produktivitas bagi manufaktur, meskipun jumlahnya tidak mendekati apa yang dihasilkan oleh sebagian besar operasi manufaktur produk, sehingga manfaatnya, meskipun nyata, tidak terlalu besar.
3. ***Modular:*** Bagian bangunan yang berfungsi penuh dan multiguna yang diproduksi di pabrik di luar lokasi. Sering kali, ini adalah kamar mandi, kamar hotel, atau bagian bangunan lain yang variasi ukuran dan fiturnya dapat dibatasi tanpa mengorbankan tujuan pemilik bangunan.

4. ***Volumetrik:*** Konstruksi modular yang terhubung sepenuhnya dengan desain, sistem, dan operasi bangunan. Ini berarti bahwa bangunan tersebut dirancang untuk menggunakan desain modular di setiap tingkat.

Tingkat-tingkat ini menjadi lebih efisien, digerakkan secara digital, dan dapat diotomatisasi saat berubah dari prafabrik menjadi volumetrik, dan kuantitas yang diasumsikan berubah dari satu kali menjadi ratusan ribu peningkatan serupa di sepanjang sumbu yang sama.

IC tidak akan membuat Anda kehilangan pekerjaan – secara sederhana, IC hanya melakukan pekerjaan Anda di luar lokasi, tetapi bahkan versi IC yang lebih rumit pun memerlukan pemahaman tentang berbagai jenis pekerjaan agar dapat berfungsi. Menurut Don Metcalf, manajer prefabrikasi di Nemmer Electric, manfaat yang ia lihat pada pekerjaan nyata meliputi:

- *Hemat waktu:* Anda sering kali dapat menyelesaikan pekerjaan lebih cepat di lingkungan yang terkendali seperti bengkel prefabrikasi daripada di lokasi kerja.
- *Fleksibilitas:* Tidak terikat dengan jadwal lokasi kerja berarti Anda dapat mengerjakan prefabrikasi saat cuaca buruk, dan menjadwalkannya saat memungkinkan karena alasan lain.
- *Biaya*: Karena lebih cepat, produksinya lebih murah. Ada juga peluang untuk membeli bahan-bahan yang umum digunakan dalam jumlah besar, seperti saluran atau lembaran logam yang dibutuhkan perusahaan untuk beberapa pekerjaan, yang selanjutnya akan menurunkan biaya.
- *Kualitas:* Melakukan hal yang sama di tempat yang sama berarti Anda dapat melakukannya dengan lebih baik. Juga jauh lebih mudah untuk memeriksa pekerjaan yang dilakukan di luar lokasi.

Konstruksi prefabrikasi dan industri juga dapat mengancam kebanggaan atas pekerjaan yang dilakukan dengan baik, yang merupakan bagian dari menjadi seorang profesional konstruksi, karena membangun dan memasangnya tidak terasa sama seperti mengerjakannya di lokasi. Seiring berjalannya waktu, rasa bangga itu akan mencakup hal-hal lain, seperti merancang sub-rakitan yang lebih baik, menemukan cara yang lebih baik untuk membuat, dan berbagai jenis masalah lain yang merupakan inti dari kebanggaan atas pekerjaan yang dilakukan dengan baik. Para manajer sebaiknya mengingat bahwa uang bukanlah satu-satunya alasan orang pergi bekerja setiap hari, dan menyadari bahwa keinginan untuk menciptakan sesuatu itu penting.

Memahami keempat tingkatan IC ini mengarah pada pertanyaan berikutnya – bagaimana hal ini sesuai dengan proses desain dan pembangunan yang mencakup pengajuan, RFI, dan perintah perubahan? Bagaimana proses itu bekerja dengan proses desain yang semakin mirip dengan teknik industri, di mana cara Anda membuat sesuatu merupakan bagian penting dari desain seperti halnya apa yang Anda buat? Konstruksi modern pada dasarnya memiliki dua kelompok desainer: arsitek yang dikenal sebagai profesional desain, kemudian pekerja terampil yang merancang sub-rakitan secara formal, dan secara informal merancang pekerjaan spesifik yang akan dilakukan. Pada dasarnya, pekerja terampil seperti insinyur

manufaktur industri yang harus mencari tahu cara membuat desain yang mereka terima menjadi sesuatu yang dapat dibangun.

Ada frasa dari manufaktur yang akan beresonansi dengan siapa pun yang pernah berkecimpung di bidang ini, dan harus mencoba membangun sesuatu yang tidak berhasil: "over the wall," yang merujuk pada cara tertua dan paling tidak efisien untuk merancang dan memproduksi suatu produk. Pemikiran semacam ini menyebabkan semacam revolusi yang meminjam dari gerakan produksi ramping: desain untuk manufaktur dan perakitan, atau DFMA.

DFMA

DFMA sebenarnya tentang dua hal – merancang produk agar dapat diproduksi, dan kemudian merancangnya agar produk tersebut dapat dirakit secara efisien di lokasi kerja. Banyak pembaca yang pernah membaca statistik tentang seberapa besar biaya pekerjaan yang ditentukan oleh keputusan di berbagai tahap – dengan lebih dari 70% biaya pekerjaan diputuskan dalam fase desain. Saat itulah hal-hal seperti atap, luas kaki persegi, jumlah lantai, dan bentuk keseluruhan bangunan diperhitungkan.

Hal yang sama berlaku saat kita merancang salah satu level IC, tetapi terutama modular dan volumetrik, di mana kuantitasnya akan cukup besar sehingga benar-benar memikirkan cara memproduksi komponen atau rakitan akan berdampak besar pada biaya, efisiensi, dan kualitas. Sama seperti di lokasi kerja, kita perlu memikirkan alat apa yang akan dibutuhkan untuk memproduksi rakitan; pada kenyataannya, seluruh proses desain lebih menyerupai pabrik mobil daripada konstruksi. Ada perangkat lunak yang dapat membantu proses ini; pada kenyataannya, istilah "DFMA" merupakan merek dagang Boothryd Dewhurst, dan mereka menawarkan perangkat lunak yang secara eksplisit ditujukan untuk mengurangi biaya dari proses produksi dan perakitan.

Desain Untuk Produksi

Karena Anda menciptakan sesuatu yang akan diproduksi berulang kali, ada baiknya untuk memikirkan setiap bagian dari proses, setiap elemen dari apa yang sedang dibuat, dan bertanya di mana kita dapat menemukan efisiensi. Daftar sebagian tempat efisiensi dapat ditemukan meliputi:

1. *Bahan* - apakah kita memerlukan jenis logam tertentu, ketebalan pipa, dan sebagainya? Apakah bahan tertentu lebih mudah dibuat dalam skala besar daripada yang lain? Apakah kita memiliki pilihan di bengkel produksi yang tidak akan kita miliki di lapangan, karena kita memiliki mesin yang lebih baik/lebih besar/lebih bertenaga?
2. *Waktu siklus* - dapatkah kita merancang untuk waktu siklus yang lebih pendek?
3. *Perkakas khusus* - seberapa banyak proses yang dapat menggunakan perkakas siap pakai?
4. *Tenaga kerja langsung* – berapa banyak waktu yang dihabiskan untuk menyentuh benda kerja?
5. *Tenaga kerja tidak langsung* – berapa banyak waktu yang dihabiskan untuk mendukung proses produksi?
6. *Limbah* – berapa banyak material yang terbuang?

7. *Barang yang mudah rusak* – mata bor, roda gerinda, dan sebagainya.

Dengan mempertimbangkan setiap elemen ini secara saksama dan membuat kompromi seperlunya, nilai dapat dimaksimalkan bagi pemilik bangunan dan kontraktor. Namun, agar ini terjadi, proses rekayasa perlu melibatkan semua pemangku kepentingan, bukan hanya kontraktor khusus yang mengerjakannya. DFMA menyediakan kerangka kerja untuk benar-benar memeriksa biaya yang mungkin timbul dari sebuah benda kerja. Kerangka kerja berikutnya memperluas elemen-elemen ini untuk menyertakan pemangku kepentingan lainnya: DFA.

Desain untuk Perakitan

Apa pun yang kita fabrikasi atau produksi, itu perlu diangkut ke lokasi, lalu ke tempat di mana ia akan menjadi bagian dari bangunan. Itu berarti naik truk, turun truk dan melintasi lokasi kerja, lalu naik dan masuk ke lantai bangunan. Terkadang seluruh jadwal pekerjaan perlu diubah untuk mengakomodasi pemasangan bagian modular yang besar, misalnya ruang mekanik. Lebih sering, komponen modular atau rakitan prafabrikasi harus mengakomodasi jalur yang akan ditempuh untuk mencapai lokasi kerja.

Pertimbangannya meliputi:

1. Keterampilan pekerja di lokasi: Tidak semua orang tahu cara memasang unit prafabrikasi. Unit tersebut harus dirancang agar sesuai dengan keahlian yang diharapkan, atau mudah dipelajari dengan cepat.
2. Jumlah pekerja di lokasi: Perakitan yang lebih besar mungkin berarti lebih banyak anggota tim.
3. Peralatan yang dibutuhkan untuk merakit atau memasang: Apakah diperlukan perangkat khusus? Atau dapatkah dilakukan dengan peralatan standar?

Pertimbangan Lain tentang DFMA

Tujuan umum DFMA adalah merancang pekerjaan untuk menurunkan keseluruhan biaya dan kompleksitas produksi, pengangkutan, pemasangan, dan kepemilikan pekerjaan yang akan dilakukan. Itu harus mencakup seberapa mudah pekerjaan tersebut dirawat, seberapa besar dampak lingkungan yang telah kita kurangi, dan seberapa mudah memeriksa kualitas dan keausan seiring berjalannya waktu.

DFMA adalah keterampilan baru yang mencapai kematangan dalam konstruksi, meskipun berbagai jenis bangunan dan pekerjaan dalam konstruksi berarti ini akan menjadi area untuk pengembangan di tahun-tahun mendatang.

5.2 ALAT DAN PROSES INFORMATION CONSTRUCTION

Memproduksi sesuatu di fasilitas manufaktur berarti Anda memiliki opsi untuk proses produksi yang tidak mungkin dilakukan di lokasi kerja. Berikut adalah beberapa hal yang perlu dipahami:

1. CNC (*Computer-Numeric-Control*) adalah pemotongan, pengeboran, dan fabrikasi benda kerja yang digerakkan oleh mesin. Ini sering dilakukan pada bentuk sederhana seperti lembaran datar, batang, dan pipa.

2. Percetakan 3D menggunakan plastik atau logam yang diendapkan dalam beberapa lapisan untuk menciptakan bentuk yang unik. Masalah kualitas awal telah digantikan oleh mesin yang sangat bagus sehingga roket orbital dibuat dengan mesin cetak 3D. Percetakan 3D telah digunakan untuk komponen khusus lebih banyak daripada penggunaan lainnya, tetapi seiring waktu akan menemukan lebih banyak aplikasi.
3. Robotika telah digunakan dalam manufaktur selama bertahun-tahun, tetapi kurang menjadi fitur dalam proses IC berbasis konstruksi karena biaya sebagian besar robot berarti mereka perlu memiliki volume produksi yang besar, yang saat ini tidak dapat dipenuhi oleh sebagian besar proyek IC konstruksi.
4. Lini perakitan adalah bagian paling dasar dari manufaktur, dan merupakan cara baru untuk memikirkan pekerjaan konstruksi, karena setiap orang hanya mengerjakan sebagian dari perakitan, lalu menyerahkannya kepada orang lain. Hal ini memungkinkan spesialisasi dan efisiensi yang sangat tinggi, tetapi bukan yang diinginkan oleh sebagian besar perdagangan dan dapat terjadi penolakan.

Perangkat Lunak IC

Tidak mengherankan, beberapa perangkat lunak yang berasal dari manufaktur dapat diterapkan pada konstruksi, tetapi, seperti halnya jalur perakitan, perangkat lunak tersebut benar-benar dibuat untuk volume yang jauh lebih tinggi dan kumpulan data yang lebih besar yang memungkinkan kontrol statistik yang tidak sesuai untuk pekerjaan konstruksi atau serangkaian pekerjaan.

Ada dua kelas perangkat lunak yang dapat mengelola tugas konstruksi industri: ***perangkat lunak desain*** dan ***perangkat lunak manajemen***. Di sisi desain, Revit, perangkat lunak BIM utama yang digunakan untuk sebagian besar VDC, tidak dibuat untuk pekerjaan fabrikasi, dan banyak yang berpendapat bahwa perangkat lunak tersebut tidak benar-benar dibuat untuk penggunaan tersebut. Namun, yang lain berpendapat bahwa hal itu terjadi karena terlalu banyak orang yang belum belajar menggunakan fungsi alat tersebut, tetapi itu lebih dari cukup untuk menangani fabrikasi.

Namun, untuk CNC dan kontrol mesin, Anda sebaiknya menjelajahi perangkat lunak desain khusus yang dibuat untuk itu, seperti Fusion360, Solidworks, Rhinoceros, Creo Parametric, dan AutoCAD Mechanical. Meskipun masing-masing memiliki fitur yang berbeda, semuanya dibuat untuk membuat model 3D yang kemudian dapat dikirim ke mesin fabrikasi, baik itu CNC, pencetakan 3D, atau yang lainnya.

Di sisi manajemen, ada pendatang baru yang dibuat khusus untuk mengelola berbagai tingkatan proses IC. Tiga pilihannya adalah:

- *Manufaktur* – lebih merupakan solusi manajemen proses, perangkat lunak ini memungkinkan kolaborasi, penjadwalan, dan komunikasi untuk proyek IC.
- *KitConnect* – dibuat oleh Project Frog, perangkat lunak ini menghubungkan desain dengan manufaktur hingga pengiriman. Ditujukan untuk digunakan dalam lingkungan BIM, KitConnect adalah produk yang matang.
- *FactoryOS* – seperti namanya, FactoryOS adalah alat manajemen untuk pabrik yang lengkap, dengan fokus khusus pada pembangunan rumah.

Konstruksi industri telah membuat langkah besar dalam beberapa tahun terakhir, dan seiring kita keluar dari karantina wilayah akibat Covid-19, diharapkan pengaturan pabrik akan memungkinkan perekrutan pekerja milenial yang sulit dipahami, dan memungkinkan produktivitas yang dibutuhkan untuk membangun dunia tahun 2020-an dan seterusnya.

BAB 6
PEMBELAJARAN MESIN DAN KECERDASAN BUATAN

"Kami mengusulkan agar dilakukan penelitian kecerdasan buatan selama 2 bulan dengan 10 orang selama musim panas tahun 1956 di Dartmouth College di Hanover, New Hampshire. Penelitian ini akan dilakukan berdasarkan dugaan bahwa setiap aspek pembelajaran atau fitur kecerdasan lainnya pada prinsipnya dapat dijelaskan dengan sangat tepat sehingga mesin dapat dibuat untuk menirunya. Upaya akan dilakukan untuk menemukan cara membuat mesin menggunakan bahasa, membentuk abstraksi dan konsep, memecahkan berbagai jenis masalah yang sekarang hanya dapat dilakukan oleh manusia, dan meningkatkan kemampuan mereka sendiri. Kami pikir kemajuan yang signifikan dapat dicapai dalam satu atau beberapa masalah ini jika sekelompok ilmuwan yang dipilih dengan cermat mengerjakannya bersama-sama selama musim panas."

J. McCarthy, 1955

Seperti yang ditunjukkan kutipan di atas, kecerdasan buatan (AI) bukanlah ide baru; faktanya, kita telah berusaha membuatnya berhasil selama hampir 70 tahun. Dan lebih dari sekali, seperti sekarang, orang mengira AI akan menjadi seperti manusia dalam kemampuannya. Polanya selalu sama setiap saat – kita memiliki beberapa kemenangan awal yang benar-benar mengesankan, dan para peneliti serta media semuanya berpikir kemajuan awal ini akan terus berlanjut, hingga kita menciptakan AI yang sesungguhnya.

Dan tentu saja, sejauh ini belum ada yang mendekati penciptaan AI yang dapat berpikir, sebagian karena tidak ada yang benar-benar tahu seberapa sulit banyak hal yang kita sebut "kecerdasan" sebenarnya – faktanya, tidak seperti hampir semua jenis teknologi lainnya, kita tidak memahami mekanisme yang mendasarinya. Ide dasar "kecerdasan," dan hal-hal spesifik tentang bagaimana berpikir terjadi, tidak dipahami pada manusia, apalagi pada mesin yang kita ciptakan. Dalam kutipan di awal bab ini, saya menyertakan semua penulis makalah pertama itu karena dua di antaranya adalah yang terbaik dalam ilmu komputer: Marvin Minsky mendirikan laboratorium media MIT, dan Claude Shannon meletakkan dasar bagi semua komunikasi modern. Orang-orang yang sangat pintar telah meremehkan betapa sulitnya hal ini sejak awal, jadi tidak mengherankan jika hal ini terus terjadi.

6.1 APA ITU PEMBELAJARAN MESIN DAN KECERDASAN BUATAN?

Pembelajaran mesin tidak sama dengan kecerdasan buatan, dan ada upaya untuk membuat AI yang tidak menggunakan pembelajaran mesin. Faktanya, ada tiga gelombang AI: dua gelombang pertama diikuti oleh kekecewaan dan apa yang sering disebut sebagai "musim dingin AI".

Namun, hampir semua yang Anda dengar disebut sebagai "AI" saat ini memiliki pembelajaran mesin sebagai intinya, jadi mari kita buat beberapa definisi yang jelas:

Pembelajaran mesin: Studi dan pembuatan algoritme berbasis perangkat lunak yang membangun model matematika berdasarkan data, yang dapat membuat keputusan, prediksi, atau melakukan tugas tanpa diprogram secara khusus untuk melakukan tugas-tugas ini. Hal ini berbeda dengan perangkat lunak biasa dalam beberapa hal utama:

1. Pembelajaran mesin bertujuan untuk menciptakan sistem yang belajar dari pengalaman, menjadi lebih baik saat Anda menggunakannya
2. Pembelajaran mesin melatih modelnya pada data dalam jumlah besar, sering kali jutaan titik data
3. Pembelajaran mesin dapat menangani tugas yang jauh lebih rumit

Kecerdasan buatan: Perangkat lunak yang meniru beberapa aspek kemampuan mental manusia, termasuk penglihatan mesin, pemahaman bahasa alami, pengambilan keputusan, dan pengenalan pola. AI dapat dibangun menggunakan pembelajaran mesin, meskipun ada metode lain yang telah digunakan untuk mencoba meniru kemampuan manusia, seperti "sistem pakar" pada tahun 1980-an.

Cara sederhana untuk memahami keduanya adalah bahwa pembelajaran mesin adalah serangkaian teknik untuk membangun AI, dan AI adalah serangkaian aplikasi dan produk yang tersedia bagi pengguna.

Mengapa AI?

Dalam dekade terakhir, ada banyak kehebohan tentang AI. Kami secara konsisten berpikir AI adalah peluang yang lebih besar dan ancaman yang lebih besar daripada yang mungkin terjadi, dan siklus kehebohan ini pasti sampai pada titik di mana pembicara dan penulis berpikir bahwa mereka berpandangan jauh ke depan ketika mereka membuat klaim provokatif tentang bahaya dan peluang. Mari kita lihat peluang yang realistis.

AI modern dapat melakukan hal-hal yang tidak dapat dilakukan oleh perangkat lunak lain. Bahkan, dibandingkan dengan pendekatan perangkat lunak lain, AI dapat tampak hampir ajaib. AI dapat memprediksi apa yang akan terjadi selanjutnya, dapat mengenali wajah dari jutaan wajah. AI dapat menemukan pola di lautan data, dapat melindungi rekening bank Anda, dan dapat menyingkirkan email spam. AI dapat mengenali bahasa dan menghasilkan respons yang tampaknya cerdas.

Dalam hal berinteraksi dengan komputer, AI telah sepenuhnya merevolusi kemungkinan, dan dalam beberapa kasus realitas, pengalaman pengguna. Misalnya, sekarang kita memiliki antarmuka suara, antarmuka gerakan, dan perangkat lunak yang dapat menebak apa yang mungkin Anda inginkan selanjutnya. Untuk sistem yang lebih besar, seperti perangkat lunak manajemen konstruksi, Anda akan dapat menyusun dan menganalisis data Anda sehingga gambaran yang lebih besar dapat dilihat. AI menjanjikan pemberdayaan manajer untuk benar-benar memahami apa yang terjadi sekarang, dan apa yang mungkin terjadi selanjutnya, dalam bisnis mereka.

Untuk solusi yang lebih kecil, kita melihat semuanya mulai dari operasi suara di lokasi kerja hingga visi mesin yang mengidentifikasi praktik jarak sosial, hingga sistem yang dapat menggabungkan ratusan gambar menjadi rekaman lokasi kerja yang dibangun secara real-time. Aplikasi AI yang spesifik untuk masalah konstruksi ini akan terus bermunculan, seiring

dengan semakin matangnya teknologi dan semakin banyak orang yang mampu menciptakan produk baru tanpa perlu memiliki gelar doktor.

AI adalah alat yang mengasah diri sendiri – penemuan manusia yang langka yang dapat ditingkatkan dari waktu ke waktu, benar-benar dapat belajar dari penggunaan di lapangan. Itu sangat unik, sangat hebat, sehingga telah menyebabkan pikiran yang rasional mengharapkan peningkatan yang luar biasa untuk terus berlanjut, yang tidak terjadi.

Mungkin yang paling penting, karena AI adalah perangkat lunak, setelah Anda melatih model untuk melakukan sesuatu, misalnya memeriksa model BIM untuk mengetahui adanya benturan atau konstruksi, Anda dapat membuat salinan model tersebut secara gratis, artinya Anda dapat segera dan tanpa biaya membuat pasukan pemeriksa BIM. Jika manusia dipekerjakan untuk menjadi pemeriksa BIM itu, Anda sekarang memiliki sejumlah besar gaji manusia yang tidak perlu Anda bayar. Dan pasukan pemeriksa BIM tidak lagi memeriksa BIM. Setidaknya, itulah idenya.

Titik Lemah AI

Realitas praktisnya adalah bahwa AI tidak pernah sebaik itu, tidak pernah dapat diandalkan, dan tidak pernah dapat dipercaya dengan sendirinya. Manusia dengan pengalaman puluhan tahun membuat kesalahan, tetapi kita tahu mengapa mereka membuat kesalahan sehingga kita dapat memprediksi dan mengoreksi kemungkinan kesalahan. Karena AI sangat berbeda dengan pikiran manusia, kita tidak selalu dapat memprediksi kesalahan apa yang akan dibuatnya. Ini tidak berarti kita tidak dapat menggunakan AI, hanya saja kehebohannya tidak dapat dibenarkan, dan aplikasi nyata membutuhkan waktu lebih lama untuk menjadi benar daripada yang dipikirkan oleh non-pengembang. Ada dua alasan mengapa AI dapat gagal yang ingin saya sampaikan kepada Anda: Kasus-kasus ekstrem dan bias.

Kasus-kasus Ekstrem

Ingatlah bahwa kita telah membahas "kasus-kasus ekstrem" di Bab 2 – yaitu contoh-contoh ketika pengguna, atau hanya data, tidak sepenuhnya sesuai dengan apa yang Anda rancang untuk ditangani oleh perangkat lunak. Itu terjadi dengan AI, karena kita telah memasukkan sekumpulan besar data ke dalam model, tetapi data tersebut tidak akan mencakup setiap kasus yang mungkin dihadapi AI, setidaknya tidak pada awalnya. Jadi ketika kita menguji AI, kita menyadari bahwa AI tidak dapat menangani kasus-kasus ekstrem tertentu, dan kita perlu mendapatkan lebih banyak data. Kasus-kasus ekstrem merupakan masalah pada semua perangkat lunak, tetapi kasus-kasus itu dapat menjadi masalah yang lebih besar pada AI, karena kita sudah terbiasa dengan AI yang melakukan hal-hal yang tampak seperti prestasi tingkat manusia, kita berasumsi bahwa AI memiliki keluasan berpikir dan kemampuan kognitif yang sama seperti manusia, dan ternyata tidak. Dan kita dapat berasumsi bahwa demonstrasi awal yang menakjubkan akan terus meningkat pada kecepatan yang sama seperti tahap-tahap awal, dan itu tidak pernah terjadi.

Mengemudi otonom adalah contoh yang bagus untuk hal ini: kita memiliki beberapa kemenangan awal yang menunjukkan janji besar, dan banyak prediksi yang mengejutkan tentang bagaimana seluruh dunia akan menjadi tanpa pengemudi pada tahun 2019. Banyak

kekhawatiran tentang pengemudi truk yang kehilangan pekerjaan, dan sebagainya. Kecuali dunia ini penuh dengan kasus-kasus ekstrem, dan model-model tersebut masih belum dapat menangani pengemudian yang sebenarnya, terutama di jalan-jalan perumahan yang taruhannya sangat tinggi, dengan anak-anak dan keluarga dalam potensi bahaya yang jauh lebih besar daripada jalan raya.

Kasus-kasus ekstrem dan kemampuannya untuk mengalahkan bahkan model AI yang paling canggih menggambarkan ide penting dalam AI: seberapa akurat AI yang Anda butuhkan? Untuk rekomendasi buku, kami dapat menerima akurasi sekitar 80%. Namun untuk mobil yang dikendarai di lingkungan perumahan, kami memerlukan akurasi 99,9% atau lebih tinggi, dan itu sangat sulit.

Bias

Pembelajaran mesin dan produk AI yang dapat diciptakannya bergantung pada banyak sekali data. Namun, bagaimana jika data tersebut tidak mencerminkan kenyataan? Bagaimana jika Anda hanya memiliki data pria dalam kumpulan data Anda? Bagaimana jika Anda tidak memiliki cukup data dari kelompok etnis tertentu? Atau bagaimana jika Anda menyertakan data yang hanya menunjukkan pria berusia 50 tahun terlambat ke lokasi kerja, sementara pria berusia di bawah 50 tahun semuanya datang tepat waktu, hanya karena itulah yang Anda miliki dalam laporan harian Anda? AI Anda akan belajar melihat dunia dengan cara yang bias: ia tidak akan mengenali, atau salah mengenali wanita, atau minoritas yang kurang terwakili. Dan ia akan memprediksi keterlambatan pada mereka yang berusia di atas 50 tahun secara otomatis, tanpa bertanya mengapa.

Mengoreksi bias adalah salah satu alasan terbesar mengapa manusia akan dibutuhkan untuk mengelola dan mengawasi AI dalam waktu yang lama. AI tidak "berpikir", ia hanya mengorelasikan data. Jadi, kita perlu menerapkan penilaian terhadap apa yang dilakukan AI untuk memastikannya tidak salah. Kecerdasan buatan yang diterapkan saat ini melakukan beberapa hal yang menakjubkan, tetapi tidak "cerdas" dalam cara yang berarti, dan tetap saja hanya perangkat lunak. Untuk memahami apa artinya semua itu, mari kita bahas tentang bagaimana pembelajaran mesin menjalankan keajaibannya.

6.2 CARA KERJA MACHINE LEARNING

Machine learning menggunakan data untuk "mengajarkan" perangkat lunak untuk mengenali berbagai hal di dunia, berdasarkan sejumlah besar contoh. Fakta bahwa kita dapat membuat perangkat lunak yang dapat belajar merupakan hal yang lebih penting daripada yang mungkin terlihat pada awalnya, dan berbicara tentang hal yang mungkin paling penting untuk benar-benar memahami AI dan machine learning:

Daripada mencoba membandingkan machine learning dan AI dengan kecerdasan manusia, bandingkan saja dengan apa yang dapat dan tidak dapat dilakukan oleh perangkat lunak non-AI.

Machine learning adalah perangkat lunak, bukan kecerdasan.

Ada tiga pendekatan machine learning utama, yang masing-masing menangani data secara berbeda dan digunakan untuk berbagai jenis masalah. Yaitu:

1. Pembelajaran terbimbing
2. Pembelajaran tak terbimbing
3. Pembelajaran penguatan.

Memahami cara kerja ini membantu Anda memahami cara kerja machine learning di dunia nyata, dan mengapa data sangat penting bagi machine learning dan produk AI.

Pembelajaran Terbimbing

Machine learning menggunakan banyak sekali data untuk melatih modelnya. Jadi, bagaimana kita memasukkan setumpuk besar data ke dalam beberapa perangkat lunak dan berharap perangkat lunak tersebut mempelajari sesuatu? Cara termudah adalah dengan memberi label pada data tersebut, atau "diawasi." Sebagian besar data yang ada dalam model produksi AI yang nyata di dunia menggunakan pembelajaran yang diawasi, karena ini adalah cara yang jauh lebih cepat untuk melatih model.

Sebagai contoh, jika saya melatih model untuk mengenali lima ras anjing, saya akan menyiapkan sebanyak satu juta foto anjing, dengan setiap foto diberi label sebagai "Husky," "Labrador Retriever," "Bloodhound," "Bulldog," atau "Poodle." Setiap kali saya memasukkan contoh melalui model, model akan mempelajari lebih banyak tentang apa yang membuat bulldog berbeda dari pudel, karena model dapat merujuk pada label tersebut.

Kami menyebutnya "diawasi" bukan karena manusia benar-benar mengawasi pelatihan, tetapi karena model tersebut memiliki label referensi yang tepat untuk setiap titik data, dan karena seseorang di suatu tempat berpikir kedengarannya lebih baik dalam sebuah makalah akademis untuk menyebutnya "diawasi" daripada hanya "diberi label." Sejuta contoh kedengarannya banyak, tetapi pikirkan sejenak tentang seberapa banyak kita mengajarkan beberapa model, seberapa halus perbedaan yang dapat ditangani oleh produk AI di luar sana. Misalnya, Facebook, Apple, dan tentu saja Google memiliki produk yang dapat mengenali Anda, secara pribadi, dari umpan foto atau video. Dari jutaan orang yang memiliki tinggi badan, warna kulit, warna rambut, dan sebagainya, AI dapat mengenali Anda. Bandingkan dengan seseorang yang mencoba menulis aturan yang dapat mengenali Anda, alih-alih menggunakan segunung data untuk melatih model seperti yang kita lakukan dalam pembelajaran mesin. Secara teknis mungkin saja untuk menulis perangkat lunak yang dapat melakukan ini, tetapi akan sangat sulit dan membutuhkan waktu yang sangat lama. Faktanya, pada tahun 1970-an, para peneliti mencoba melakukan sesuatu seperti ini, menggunakan apa yang pada saat itu disebut "sistem pakar". Kebetulan, ini merupakan bagian dari "gelombang AI" kedua.

Para peneliti ini mencoba membuat sistem yang akan mengenali saat siswa matematika SMA berprestasi baik, atau buruk, dalam matematika. Sistem ini akan mengenali dari jawaban mereka dan isyarat lain apa yang perlu mereka dengar selanjutnya, penyegaran atau pembelajaran lain apa yang terbaik. Proyek ini ditinggalkan ketika para peneliti menyadari bahwa mereka harus memprogram lebih dari 500 jam per menit waktu siswa, atau 1.250 hari selama satu jam.

Contoh lain adalah teknologi suara. Beberapa pembaca mungkin pernah menggunakan suara Dragon untuk mengirim pesan teks di masa lalu, dan jika demikian, Anda akan ingat betapa tidak akuratnya itu, dan bagaimana ia perlu mendengar suara spesifik Anda untuk sesi pelatihan sebelum mulai berfungsi. Ada alasan mengapa kita hampir tidak mendengar tentang suara hingga sekitar tahun 2011. Sistem-sistem ini sebenarnya sangat pintar, tetapi mereka tidak belajar dari data seperti yang dilakukan pendekatan pembelajaran mesin modern.

Sebaliknya, Siri, Alexa, dan produk suara AI lainnya belajar, terus-menerus, dari ratusan juta pengguna, dan sebagai hasilnya mereka menjadi lebih baik setiap saat. Apa pun rasa frustrasi yang Anda alami dengan sistem-sistem ini, itu bukan karena mereka tidak mengerti bahasa Inggris Anda, itu karena tidak ada kecerdasan di balik mereka untuk memahami maksud Anda.

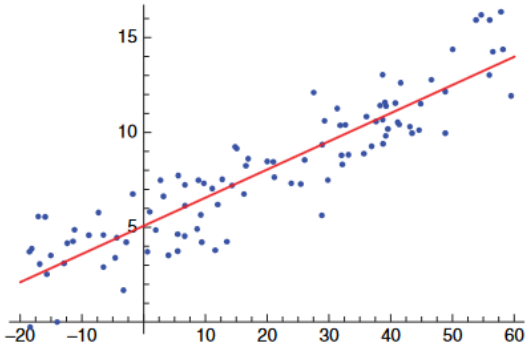
Pembelajaran dengan menghubungkan contoh-contoh ke label ini sangat mirip dengan regresi. Anda mungkin ingat dari kelas statistika gagasan sederhana bahwa jika Anda memiliki banyak titik data, dan Anda ingin memahami apakah satu titik terkait dengan yang lain, Anda memplot titik-titik ini pada grafik, di mana sumbu-x adalah satu faktor, dan sumbu-y adalah faktor lainnya.

Misalnya, jika Anda ingin tahu apakah suhu udara berdampak pada kelembapan, Anda dapat melakukan beberapa lusin pembacaan di luar ruangan, di mana Anda melihat persentase kelembapan dan suhu pada setiap pembacaan ini.

Anda akan mendapatkan beberapa lusin titik yang menunjukkan tren naik, yang menunjukkan bahwa suhu yang lebih tinggi menyebabkan kelembapan yang lebih tinggi. Itu bagus, tetapi yang benar-benar ingin Anda ketahui adalah, untuk suhu tertentu, berapa kemungkinan kelembapan yang tinggi?

Regresi adalah proses di mana Anda dapat mencari tahu apa hubungan antara kedua faktor ini, yang dinyatakan sebagai persamaan. Ini seperti pembelajaran terbimbing, di mana satu sumbu adalah label, dan sumbu lainnya adalah titik data – kecuali alih-alih dua sumbu, Anda memiliki sejumlah besar faktor.

Regresi bagus sebagai analogi sehingga Anda dapat memahami pelabelan, tetapi realitas pembelajaran mesin modern adalah bahwa matematika cukup rumit, dan untungnya sangat sedikit orang yang benar-benar membangun produk AI yang perlu benar-benar memahaminya di tingkat yang lebih dalam. Dalam praktiknya, terutama untuk gambar dan pemrosesan bahasa, Anda akan menggunakan model yang sudah ada yang telah dilatih pada jutaan contoh. Model-model ini cukup umum sehingga para insinyur kemudian dapat melatihnya kembali pada set data yang lebih kecil dan spesifik, dan para pengembang Anda dapat mengaksesnya melalui perusahaan-perusahaan besar, atau mendapatkan model sumber terbuka dari GitHub.



Misalnya, Anda dapat mengambil model pengenalan gambar dari Microsoft, Amazon, atau Google. Kemudian Anda dapat mengimpor beberapa ribu gambar, katakanlah dari sejenis alat, seperti kunci inggris. Pada awalnya, model tersebut akan memiliki akurasi yang relatif rendah, tetapi saat Anda memberinya lebih banyak contoh, model tersebut akan menjadi semakin akurat. Untuk lebih jelasnya, model tersebut tidak berjalan sendiri, seseorang perlu mengubah beberapa pengaturan, tetapi melatih model untuk penggunaan seperti ini jauh lebih layak daripada menemukan sejuta gambar kunci inggris. Tiga hal yang perlu diingat tentang pembelajaran mesin terbimbing:

1. Pembelajaran terbimbing berarti data diberi label
2. Model memerlukan jutaan contoh untuk dilatih dari awal
3. Model modern, terutama untuk gambar dan bahasa, terkadang dapat dilatih dari model AI yang ada

Artinya bagi Anda, profesional konstruksi, adalah bahwa nilai bagi organisasi Anda, dan pada akhirnya bagi Anda, dari data yang konsisten dan dipersiapkan dengan baik berpotensi sangat besar. Anda akan dapat membangun model Anda sendiri untuk memprediksi kemungkinan masalah keselamatan, pengerjaan ulang, RFI, dan banyak masalah kritis lainnya di masa mendatang, tetapi sistem yang dapat melakukan itu akan memerlukan data Anda untuk dilatih.

6.3 PEMBELAJARAN TANPA PENGAWASAN

Terkadang Anda mendapatkan data dalam jumlah besar dan Anda bahkan tidak tahu cara memberi labelnya, karena Anda tidak tahu di keranjang mana data tersebut harus ditempatkan. Contohnya adalah ketika Anda tidak tahu apa yang menyebabkan kegagalan mekanis, tetapi Anda memiliki data sensor dari berbagai bagian mesin dan sistem. Anda perlu memasukkan informasi ini, biasanya jutaan titik data, untuk mulai memahami data sensor mana yang berkorelasi dengan data sensor lainnya, sehingga Anda dapat mulai membuat label yang, pada gilirannya, akan memungkinkan Anda merancang sistem yang dapat memprediksi kegagalan ini saat kemungkinan terjadi.

Itulah yang dilakukan pembelajaran tanpa pengawasan. Perangkat lunak akan mengambil data dalam jumlah besar, dan akan mencari tahu cara membuat kelas, atau keranjang, untuk data tersebut. Hal semacam ini sebenarnya jarang digunakan untuk sistem AI produksi, tetapi justru merupakan cara untuk merancang model yang sangat besar. Misalnya, pada tahun 2018 Google tengah mengerjakan cara baru untuk melatih model agar memahami bahasa. Model tersebut disebut "Pemrosesan Bahasa Alami," atau NLP. Sebagai Google, mereka memiliki lebih banyak data daripada siapa pun, dan memiliki daya pemrosesan yang lebih besar daripada siapa pun.

Google menggunakan pembelajaran tanpa pengawasan untuk mengambil kumpulan data yang sangat besar ini, mengolahnya menjadi beberapa kelompok, dan melatih model BERT-nya untuk mengenali dan memanfaatkan kalimat bahasa yang kompleks dalam lebih dari 70 bahasa. Ilmuwan data Google telah menemukan pendekatan yang ingin mereka ambil, mereka memiliki arsitektur. Namun, mereka tidak mengetahui "kelompok", mereka tidak tahu cara memberi label pada sejumlah besar informasi, jadi mereka membiarkan sistem melakukan pengelompokan untuk mereka.

Diperkirakan Google menghabiskan sekitar Rp. 200 Miliyar hanya untuk melatih model ini, menggunakan server mereka sendiri, yang menunjukkan betapa besarnya upaya pembelajaran tanpa pengawasan.

Catatan Tambahan tentang Model AI

Pada contoh sebelumnya, tim Google memiliki "pendekatan", yang dikenal dalam AI sebagai model. Model adalah konsep penting, yang terlewatkan dalam diskusi populer tentang pembelajaran mesin dan AI. Model yang dipilih, algoritme yang menjadi bagiannya, adalah sebagian besar sumber keajaiban sebenarnya. Seperti yang telah kita lihat, data dalam jumlah besar diperlukan agar pembelajaran mesin berfungsi, terutama untuk tugas yang lebih menantang seperti visi mesin, pemahaman bahasa alami, dan lain-lain.

Dalam kasus BERT, Google menyempurnakan serangkaian jenis model, dengan nama-nama yang menarik seperti Word2Vec, ELMo, dan ULMFit, dan kemudian disempurnakan oleh GPT-3 milik OpenAI. Masing-masing model ini seperti versi mesin mobil – pada dasarnya cara kerjanya sama, tetapi komponennya ditingkatkan saat teknisi belajar dan mencoba hal-hal baru.

Kita hanya dapat melatih AI untuk melakukan hal-hal yang dimungkinkan oleh modelnya, dan itu penting untuk pemahaman kita tentang AI secara keseluruhan, dan untuk mengenali batasan implementasi atau produk AI apa pun yang akan Anda hadapi. Pembelajaran mesin, dan karenanya AI, tidak dapat melakukan apa pun yang tidak dirancang untuk dilakukannya. Pembelajaran mesin tidak dapat mempelajari keterampilan yang sama sekali baru yang tidak dirancang untuknya, pembelajaran mesin tidak dapat menciptakan kemampuan baru. Pers populer sering kali mengabaikan poin ini, seperti halnya dengan poin yang lebih besar tentang betapa sulitnya membuat model nyata benar-benar berfungsi. Melatih model seukuran BERT, misalnya, akan memakan waktu ratusan jam kerja bagi ilmuwan data untuk menyempurnakan model saat sedang dilatih dan dilatih ulang. Pekerjaan ini sulit dan melelahkan karena pada dasarnya Anda membangun, menguji, gagal, lalu membangun kembali untuk mengulang siklus tersebut hingga berhasil mencapai tingkat akurasi yang Anda perlukan.

6.4 PEMBELAJARAN PENGUATAN

Meskipun pembelajaran terbimbing adalah pekerja keras pembelajaran mesin yang digunakan di sebagian besar produk AI produksi, dan pembelajaran tak terbimbing adalah cara kita mengeksplorasi masalah data yang jauh lebih besar, lebih sulit, dan tak terstruktur,

mungkin pendekatan pembelajaran mesin yang paling menarik dalam beberapa tahun terakhir adalah pembelajaran penguatan.

Pembelajaran terbimbing dan pembelajaran tak terbimbing bagus dalam menangani proses satu kali. Tidak ada ingatan tentang apa yang telah terjadi di masa lalu, tidak ada cara untuk memiliki urutan langkah, misalnya. Pembelajaran terbimbing dan tak terbimbing sempurna untuk tugas satu kali seperti mengenali foto, menganalisis pindaian 3D, memahami paragraf, dan sebagainya. Masing-masing terjadi sekali, lalu sistem disetel ulang dan dihapus bersih. Jika saya menunjukkan gambar pengenal gambar, misalnya, tumpukan kerikil setinggi 5 kaki, lalu gambar tumpukan kerikil yang sama setinggi 7 kaki, lalu lagi setinggi 10 kaki, algoritme pengenalan gambar tidak akan melakukan hal yang berbeda setiap kali mengenali tumpukan kerikil. Fakta bahwa hal itu terjadi secara berurutan bukanlah informasi yang memiliki tempat bagi model tersebut – hal itu tidak relevan. Banyak hal yang kita inginkan dari perangkat lunak memerlukan serangkaian langkah.

Seseorang melakukan sesuatu, hal itu menyebabkan perubahan di dunia, lalu mereka melakukan hal lain, atau orang lain melakukan sesuatu sebagai respons terhadap tindakan orang pertama, dan seterusnya.

Pembelajaran penguatan disebut demikian karena sistem AI diberi hadiah setiap kali berhasil melakukan satu langkah dalam suatu proses dengan benar, sehingga kita dapat melatih langkah-langkah individual dalam suatu rangkaian. Algoritme tidak mendapatkan "hadiah," dalam pengertian apa pun yang kita kenali, itu hanyalah serangkaian langkah matematika. Kami menjalankan algoritme ratusan dan ribuan kali, setiap kali memberi penghargaan pada setiap sublangkahnya sedikit jika algoritme bekerja lebih baik, dan lebih sedikit jika tidak ada peningkatan.

Seiring dengan semakin kuatnya komputer dan kemampuan kita untuk mensimulasikan berbagai proses, kita dapat menggunakan pembelajaran penguatan dengan mensimulasikan suatu proses, dan melatih ratusan ribu versi algoritme hingga kita menemukan versi yang paling berhasil.

Kekuatan pendekatan ini menjadi jelas pada bulan Maret 2016 ketika AlphaGo milik Google mengalahkan Lee Sedol, juara dunia 18 kali dalam permainan strategi "Go." Diciptakan di Tiongkok lebih dari 2.500 tahun yang lalu, Go adalah permainan papan seperti catur yang dimainkan di seluruh Asia Timur, dan dianggap sebagai permainan tersulit yang ada.

Pembelajaran penguatan masuk akal di sini, karena untuk menang Anda perlu melakukan serangkaian gerakan dengan benar, sebagai respons terhadap apa yang dilakukan lawan Anda. Banyaknya kemungkinan gerakan dan kompleksitas Go yang luar biasa menjadikannya tantangan tersendiri.

Papan Go terdiri dari kotak-kotak berukuran 19x19, dan setiap pemain memiliki bidak hitam atau putih yang semuanya memiliki peringkat yang sama, tidak seperti dalam catur barat. Tujuan permainan ini adalah untuk merebut wilayah dengan mengepung dan menyingkirkan bidak pemain lain. Aturannya sederhana dan berhubungan dengan berapa banyak gerakan, giliran yang harus diambil, dan sebagainya. Karena aturannya sederhana dan semua bidaknya sama, permainan ini jauh lebih rumit daripada catur Barat – sebagai ilustrasi,

diperkirakan ada 10 × 10170 cara berbeda untuk menyusun papan – jumlah yang sangat besar yang sebenarnya lebih besar daripada jumlah atom di alam semesta yang dikenal. Go adalah permainan yang sulit dikuasai.

Jumlah kemungkinan yang sangat besar itu berarti Anda tidak bisa sekadar melakukan pencarian komputer untuk semua kemungkinan langkah berikutnya dan langkah-langkah selanjutnya; Anda harus mampu memahami permainan pada tingkat yang lebih dalam. Ketika Deep Blue milik IBM mengalahkan juara catur dunia, Gary Kasparov, pada tahun 1997, ia tidak menang dalam permainan catur, ia menang dalam mencari kemungkinan langkah dengan kecepatan kilat dan menemukan kecocokan terbaik. Pendekatan itu tidak berhasil untuk Go, itulah sebabnya kebanyakan orang mengira butuh waktu puluhan tahun sebelum AI dapat mengalahkan manusia, tetapi berhasil mengalahkannya. Pembelajaran penguatan tidak mudah dilakukan dengan benar, tetapi ini menunjukkan jalan menuju AI yang membantu segala hal mulai dari penjadwalan hingga otomatisasi tugas-tugas tertentu oleh perangkat lunak. Ini tidak berarti bahwa apa yang dilakukan manusia dapat diotomatisasi semudah itu. Kita akan membahasnya lagi di bawah dan di bab berikutnya.

6.5 PEMBELAJARAN MENDALAM

AI telah melalui tiga gelombang. Gelombang pertama menggunakan kumpulan pendekatan perangkat lunak untuk mendekati AI sebagai masalah logika – yang sering disebut sebagai GOFAI, atau “AI Jadul yang Baik.” Kami masih merujuk pada banyak ide dan terobosan dari periode ini, karena mereka membantu membingkai masalah, bahkan mengusulkan beberapa ide yang kemudian terbukti penting. Namun, matematika, ilmu komputer, dan yang terpenting, perangkat keras dan perangkat lunak komputer yang sebenarnya saat itu tidak cukup baik untuk mengembangkan ide-ide ini, sehingga bidang ini gagal pada tahun 1970-an.

Pada akhir tahun 1970-an dan 80-an, pendekatan lain dicoba – alih-alih beroperasi pada tingkat logika yang lebih tinggi, para peneliti mencoba menetapkan aturan. Dikenal sebagai "sistem pakar", pendekatan ini menunjukkan kemajuan awal hingga dihadapkan dengan kompleksitas aplikasi nyata. Sistem pakar gagal karena Anda tidak dapat menulis cukup banyak aturan agar perangkat lunak dapat menangani dunia nyata. Akhir tahun 1980-an melihat kegembiraan terhadap AI kembali menurun.

Namun selama ini, ada ilmuwan komputer yang berupaya keras untuk menemukan pendekatan yang berbeda, dan salah satunya, pembelajaran mendalam, ternyata penting.

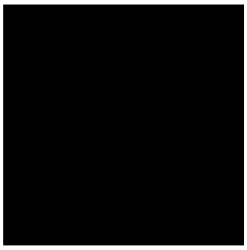
Definisi Pembelajaran Mendalam

Pembelajaran mendalam secara umum didasarkan pada cara kerja otak manusia, jadi mari kita pahami secara singkat apa artinya. Otak manusia terdiri dari sekitar 85 miliar neuron. Masing-masing neuron memiliki banyak masukan, yang disebut dendrit, yang terhubung ke banyak keluaran neuron lain yang disebut akson. Setiap neuron hanya memiliki satu akson, yang pada gilirannya terhubung ke banyak dendrit neuron lainnya. Ketika cukup banyak dendrit dari neuron tertentu menerima cukup banyak sinyal dari mitra neuron hulu mereka, neuron itu akan "mengeluarkan" sinyal, yang kemudian akan turun ke sepanjang akson, dan pada gilirannya mengirimkan sinyal ke banyak neuron lainnya.

Neuron yang khas terhubung ke rata-rata 5.000 neuron lainnya, dan sebanyak 50.000. Jadi otak adalah jaringan neuron besar yang saling menembak. Hal penting yang perlu dipahami adalah gagasan tentang banyaknya neuron yang mengirimkan sinyal ke satu neuron, dan jumlah sinyal tersebut harus melewati ambang batas agar neuron tersebut aktif. Ambang batas tersebut dicapai dengan dua cara.

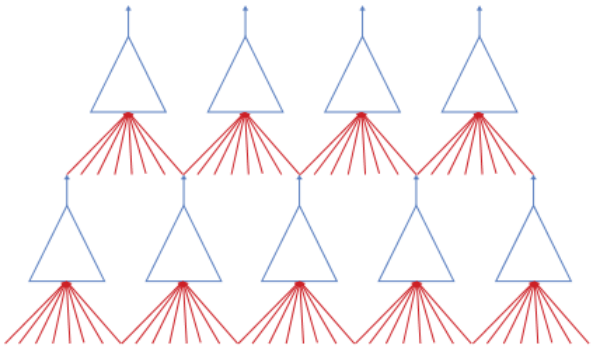
Banyaknya neuron yang mengirimkan sinyal ke neuron, dan melewati ambang batas. Atau lebih sedikit neuron yang mengirimkan sinyal ke neuron, tetapi masing-masing neuron mengirimkan sinyal yang lebih kuat, dan melewati ambang batas.

Pembelajaran terjadi karena neuron yang aktif bersama-sama sering kali koneksinya akan diperkuat, sehingga mereka dapat mengirimkan sinyal yang lebih kuat. Neuron yang mengirimkan sinyal ke neuron lain ini menciptakan hierarki, dan hierarki tersebut berlanjut di seluruh otak. Dengan kata lain, kita memiliki lapisan neuron yang berawal dari "dasar" apa pun yang kita lihat, dengar, atau pikirkan, hingga ke tingkat persepsi atau pemikiran yang lebih tinggi.



Mari kita ilustrasikan hal itu dengan contoh bagaimana otak Anda melihat sebuah kotak – langkah-langkah berbasis neuron yang memungkinkan mata Anda mengirimkan data ke otak yang diproses sehingga Anda melihat kotak itu. Kedengarannya sederhana, tetapi sebenarnya sangat canggih. Ketika mata Anda melihat kotak hitam seperti ini:

Beberapa neuron yang merespons warna hitam akan meneruskan informasi tentang apakah mata melihat warna hitam pada titik tertentu ke kumpulan neuron lain. Neuron berikutnya akan mengambil bintik hitam dan "bintik" kosong dan menjumlahkannya, sehingga beberapa neuron akan memproses tepi, beberapa akan memproses panjang yang panjang pada sudut 90 derajat, yang lain pada sudut nol derajat, dan seterusnya. Ternyata di dalam otak terdapat kelompok neuron yang pandai mengenali bentuk dasar, tepi, dan sebagainya. Kelompok ini dilatih untuk melihat bentuk dan hubungan, sehingga mereka aktif saat melihatnya.



Selanjutnya, neuron ini meneruskan sinyalnya ke neuron yang mengenali sudut tempat dua panjang terhubung, kemudian neuron ini akan meneruskan sinyalnya ke neuron yang mengenali kotak, dan seterusnya hingga kotak dikenali. Setiap lapisan ini telah dilatih selama bertahun-tahun untuk mengenali hal-hal yang dilihatnya di dunia, dengan melihatnya berulang-ulang. Itulah yang dilakukan pembelajaran mendalam.

Dalam pembelajaran mendalam, sejumlah "simpul" dibuat dalam perangkat lunak yang bekerja seperti neuron manusia. Mereka mengambil masukan dari sesuatu seperti kamera, dan meneruskannya ke sejumlah simpul ini. Berdasarkan kekuatan yang dimiliki

masing-masing simpul ini dengan sumber masukan, beberapa di antaranya aktif dan meneruskan informasi ke lapisan berikutnya, dan beberapa tidak. Ini terjadi beberapa kali, sering kali tidak lebih dari 15 lapisan, lalu sebuah keluaran dibuat. Biasanya keluaran itu mengklasifikasikan sesuatu, entah itu gambar anjing, kalimat, atau pola dalam data keuangan.

Kekuatan pembelajaran mendalam adalah, setiap kali Anda memberinya masukan yang memiliki label, ia dapat "belajar" karena klasifikasi yang salah akan menyebabkan seluruh jaringan menyetel ulang kekuatan di antara simpul-simpul ini.

Jadi katakanlah, misalnya, Anda memiliki tiga lapisan dalam jaringan, dan kami memilih tiga neuron dari lapisan 2 yang masuk ke satu neuron di lapisan ketiga. Selama pelatihan, gambar seekor anjing diproses, tetapi jaringan mengira itu adalah manusia. Itu diproses sebagai kesalahan, dan koneksi antara neuron 1, 2, dan 3 akan diubah. Yang membantu tentang cara ini dilakukan dalam praktik adalah bahwa pelatihan akan mengetahui ke arah mana kekuatan perlu diubah – lebih kuat atau lebih lemah. Seberapa banyak kekuatan itu diubah ditentukan oleh ilmuwan data yang melakukan pelatihan.

Proses ini terjadi berulang-ulang hingga model cukup baik dalam tugas yang Anda latih – sering kali pelatihan dapat berlangsung ratusan atau ribuan siklus ini. Karena kami telah membagi masalah mengidentifikasi sesuatu ke dalam hierarki seperti ini, kami dapat menjadi sangat fleksibel, dan juga sangat kuat. Itulah keajaiban pembelajaran mendalam, dan mengapa itu berada di balik banyak hal yang berhasil dalam gelombang ketiga AI ini.

6.6 IMAGENET DAN KELAHIRAN AI MODERN

Yang luar biasa adalah bahwa pendekatan pembelajaran mendalam telah dikenal selama beberapa dekade, tetapi kami tidak cukup mampu membuatnya bekerja seperti sekarang. Tiga perubahan telah memungkinkan pembelajaran mendalam: lebih banyak data, komputer yang lebih baik, dan model yang lebih baik.

Big Data

Internet, World Wide Web, dan telekomunikasi telah mengubah cara manusia mengonsumsi, memproduksi, dan berbagi informasi. Coba pikirkan gambar – yang dulu memerlukan kamera khusus, film khusus, dan setidaknya proses semalam untuk mengembangkan gambar kini dapat dilakukan dengan mudah, langsung dari ponsel Anda. Demikian pula, dulu memerlukan penyalinan yang mahal dan surat fisik untuk berbagi gambar, kini kita mengirim gambar melalui email, pesan, media sosial, dan banyak cara lain yang kita anggap biasa.

Penciptaan, penyalinan, dan pengiriman yang mudah ini berlaku untuk hampir semua jenis informasi. Akibatnya, penciptaan data di seluruh dunia begitu besar sehingga sekali lagi melibatkan angka-angka yang tidak dapat kita pahami sebagai manusia. Secara pribadi, saya menyukai nama-nama yang digunakan – menurut Forum Ekonomi Dunia, umat manusia telah menciptakan sekitar 44 zettabyte data pada tahun 2020. Bagaimana kita mencapai zettabyte? Jika Anda memiliki sejuta byte untuk membuat satu megabyte (MB), 1.000 kali lipatnya adalah gigabyte (GB), dan 1.000 kali lipatnya adalah satu terabyte (TB) – semua angka yang mungkin pernah Anda gunakan untuk file, RAM komputer, dan penyimpanan hard drive, masing-

masing. Seribu terabyte adalah satu petabyte, yang merupakan ukuran data yang dihasilkan Facebook dalam sehari (sekitar 4 PB pada tahun 2019).

Seribu petabyte adalah satu exabyte, jenis kuantitas yang diciptakan dunia dalam satu hari, diperkirakan mencapai 265 exabyte per hari pada tahun 2025. Seribu exabyte membawa Anda ke zettabyte, yang, Anda ingat, dunia memiliki sekitar 44. Namun pada tingkat ini kita akan melampaui 1.000 zettabyte, di mana kita akan sampai ke yottabyte.

Secara pribadi, saya ingin diundang ke pertemuan berikutnya di mana kuantitas data diberi nama – sepertinya menyenangkan. Yang paling tidak sama pentingnya dengan jumlah data adalah kemampuan kita untuk mengelola semua data ini. Kami telah menemukan cara baru untuk menyimpan, mencari, dan memproses data yang membuatnya lebih berguna, sering kali bertahun-tahun setelah dikumpulkan, untuk pelatihan model AI.

Daya Komputasi

Tumpukan data yang tak terbayangkan ini membutuhkan usaha untuk diproses. Peningkatan daya komputasi selama 70 tahun terakhir sudah mapan dan terjadi di sekitar kita. Banyak dari Anda pasti pernah mendengar tentang "hukum Moore," yang mengatakan bahwa daya pemrosesan chip komputer akan berlipat ganda setiap 2 tahun. Hal ini muncul dari pengamatan Gordon Moore, salah satu pendiri Intel, pada tahun 1970, bahwa ukuran transistor yang menyusun sebuah microchip akan menyusut setengahnya dan biayanya juga akan berkurang setengahnya, dalam periode 2 tahun tersebut. Hasil praktisnya adalah daya microchip akan berlipat ganda setiap 2 tahun.

Fold	Tebal	Satuan
1	0.0625	inchi
2	0.125	
3	0.25	
4	0.50	
5	1.00	
6	2.00	
7	4.00	
8	8.00	
9	1.30	Kaki
10	2.70	
11	5.30	
12	10.70	
13	21.30	
14	42.70	
15	85.30	
16	170.70	
17	341.30	
18	682.70	
19	1,365.30	
20	2,730.70	
21	1.03	Mil
22	2.06	

23	4.10	
24	8.30	
25	16.50	
26	33.10	
27	66.20	
28	132.40	
29	264.80	
30	529.60	
31	1,059.20	Mil
32	2,118.30	Mil
33	4,236.70	Mil
34	8,473.30	Mil
35	16,946.70	Mil
36	33,893.40	Mil
37	67,786.70	Mil
38	135,573.50	Mil
39	271,146.90	Mil

Kedengarannya keren, dan penggandaan itu mengesankan. Daya penggandaan yang konstan adalah bahwa ia akan bertambah banyak. Mungkin, misalnya, Anda pernah mendengar eksperimen pikiran tentang menggandakan selembar kertas. Pertanyaan inti dari eksperimen tersebut adalah: berapa kali Anda harus melipat selembar kertas untuk mencapai bulan?

Tidak sebanyak yang Anda kira! Jika Anda melipat selembar kertas menjadi dua, menggandakan ketebalannya, Anda mungkin akan mendapatkan 1/16 inci (#1). Gandakan lagi (#2), dan Anda akan mendapatkan 1/8 inci (#3) – masih cukup tipis. Namun perhatikan apa yang terjadi saat Anda terus menggandakannya – hanya dengan 39 kali penggandaan, kertas Anda yang sederhana akan lebih tebal dari jarak ke bulan.

Jarak ke bulan adalah 238.855 mil, yang berarti hanya dalam 39 kali lipat kita sudah sampai di sana!

Sekarang bayangkan apa yang terjadi pada komputer. Hukum Moore pertama kali "dimulai" pada tahun 1965, yaitu sekitar 27 kali lipat – jadi kita akan berubah dari 1/16 inci menjadi lebih dari 66 mil dalam eksperimen melipat kertas kita. Itu adalah perbedaan yang tak terbayangkan, dan itu terlihat dalam kekuatan komputer saat ini. Faktanya, komputer modern telah berubah dari beberapa ratus transistor menjadi 2,9 miliar dalam Intel i7 terbaru, dengan chip khusus mencapai tingkat yang lebih tinggi. Sebagai konsumen, kita dapat menganggap hal-hal ini biasa saja, karena mereka muncul sebagai permainan yang lebih cepat, spreadsheet yang lebih besar, dan tentu saja ponsel yang melakukan hal-hal yang hampir ajaib dibandingkan dengan apa yang dapat dilakukan apa pun bahkan 10 tahun yang lalu. Bagi AI, itu berarti bahwa hal-hal yang sama sekali baru menjadi mungkin.

Algoritma

Ide dasar untuk menumpuk neuron buatan ke dalam jaringan saraf bukanlah hal baru; faktanya, Steve Jobs telah mencoba melakukan ini pada tahun 1990-an dengan komputer

"NEXT" miliknya. Saat itu, kita tidak tahu cara menangani jaringan saraf, dan pendekatannya adalah memprogramnya alih-alih melatihnya.

Pada tahun 2006, seorang peneliti di Universitas Stanford, Fei-Fei Li, membuat sekumpulan data untuk melatih AI dalam mengenali gambar, yang disebut ImageNet. Data ini menjadi dasar kompetisi tahunan di Stanford, di mana setiap tahun pendekatan yang berbeda dicoba untuk melakukan pekerjaan yang lebih baik dalam mengklasifikasikan data. Dan setiap tahun pendekatan tersebut menjadi sedikit lebih baik.

Kemudian Profesor Geoff Hinton dan timnya dari Universitas Toronto membawa model pembelajaran mendalam ke ImageNet, dan menunjukkan peningkatan sebesar 10%, mengalahkan tim mana pun sebelumnya. Seperti yang dilakukan para akademisi, ia menerbitkan pendekatannya, dan sisanya adalah sejarah. Hampir semua AI yang Anda gunakan sekarang memiliki pembelajaran mendalam sebagai intinya, dan dua hal yang perlu Anda ingat adalah bahwa 95% AI berkaitan dengan data, dan tingkat akurasi yang Anda perlukan.

Melatih AI

Sebelumnya, kita melihat bagaimana regresi menjadi analogi yang baik untuk cara kita melatih data dalam pembelajaran terbimbing – Anda menandai semua data Anda dan seiring waktu komputer mempelajari bahwa hal-hal yang tampak seperti data Anda berkorelasi dengan apa pun labelnya.

Jadi, jika setiap kali suhu naik 1 derajat, kelembapan naik sekitar 2%, Anda memiliki persamaan sederhana 1 suhu = 2 kelembapan. Jika suhu 85 derajat dan kelembapan 50%, dan suhu naik 5 derajat, saya seharusnya memiliki kelembapan 60%. Cukup sederhana.

Kecuali dunia tidak bekerja seperti itu, tentu saja tidak sepanjang waktu. Masalah dengan penggunaan data untuk menciptakan hubungan semacam ini adalah dunia ini berantakan. Jadi, hubungan apa pun yang dapat kita cari tahu hanya akan mencakup sebagian kecil dari dunia nyata, dan kita memiliki cara untuk mengukur keakuratan ini – dalam kasus regresi, kita menyebutnya R-kuadrat.

Mungkin hal terpenting yang perlu Anda pahami dari bab ini adalah bahwa tidak ada pelatihan yang mencakup segalanya, dan tidak ada komputer yang mampu beroperasi di luar apa yang telah Anda latih untuk dilakukannya. Dijamin bahwa jika Anda menggunakan sistem AI cukup lama, Anda akan menemukan hal-hal yang tidak dapat ditanganinya.

Kita melatih AI agar ia belajar mengenali dan beroperasi pada contoh nyata dan pekerjaan nyata. Itu berarti kita membutuhkan banyak data, dan data tersebut perlu dipersiapkan dengan saksama agar Anda tidak melatih AI Anda untuk mengenali hal yang salah.

Contoh yang bagus datang dari ceramah TEDx Dirigo1 yang disampaikan oleh Peter Haas, seorang peneliti robotika, pada tahun 2017. Haas menceritakan bagaimana mereka melatih sistem untuk mengenali ras hewan, dan menemukan bahwa AI mereka salah mengklasifikasikan anjing husky sebagai serigala. Kebanyakan orang akan membayangkan ini adalah kesalahan yang wajar dan tidak disengaja, lagipula mereka memang sangat mirip.

Haas dan rekan-rekannya tidak puas dengan penjelasan itu, jadi mereka menulis beberapa kode untuk mencari tahu mengapa AI mengira anjing husky itu serigala, dan apa yang mereka temukan sungguh mengejutkan. AI tidak melihat mata atau moncong hewan itu. Sebaliknya, AI memperhatikan bahwa semua foto serigala itu bersalju di tanah, dan foto anjing husky itu juga bersalju di tanah, jadi tentu saja hewan di salju itu pasti serigala.

Pada tahun 2017, perlu diingat bahwa banyak orang mengira AI adalah ancaman terbesar yang dihadapi manusia, melampaui perubahan iklim, perang, dan penyakit. Dan pesan Haas adalah bahwa AI bukanlah teknologi yang sangat kompeten, tetapi merupakan teknologi yang sangat bodoh.

Kita perlu melatih data kita dengan hati-hati, memeriksa apa yang dihasilkannya, dan selalu meminta manusia untuk mengawasi karena fakta terpenting tentang AI adalah AI itu bodoh.

BAB 7
MENERAPKAN KECERDASAN BUATAN

Apakah kecerdasan buatan akan menggantikan pekerjaan Anda? Mungkin. Setelah puluhan tahun kemajuan AI, dan sekitar satu dekade penerapan AI pembelajaran mendalam generasi baru, hilangnya pekerjaan yang dijanjikan belum terwujud. Seperti teknologi lainnya, AI akan mengubah lanskap pekerjaan, tetapi berhati-hatilah untuk tidak mempercayai sensasi tentang seberapa besar, dan seberapa cepat AI akan menggantikan apa yang dilakukan manusia – jauh lebih sulit untuk menerapkan teknologi AI di lapangan daripada mendemonstrasikannya di sebuah konferensi. Mengapa khawatir? Karena kita telah melihat industri AS seperti manufaktur, baja, bahkan pertanian, kehilangan banyak pekerja di masa lalu, dan sepertinya AI mungkin menjadi gelombang berikutnya, yang akan menggantikan konstruksi serta sektor lainnya – tetapi apakah sesederhana itu?

Sektor manufaktur telah mengalami perubahan besar dalam 50 tahun terakhir, dengan pabrik baja, pabrik mobil, dan seluruh sektor seperti tekstil diotomatisasi atau dikirim ke luar negeri. Kenyataannya adalah bahwa ini adalah gambaran yang sangat rumit, tetapi telah menghasilkan ketakutan yang mengakar terhadap otomatisasi pada pekerja di berbagai industri.

Dari sekitar tahun 2015 hingga beberapa waktu lalu, laporan demi laporan meramalkan bahwa sejumlah besar pekerjaan akan digantikan oleh AI – banyak yang dengan gembira menunjukkan bahwa pekerjaan kerah putih terancam untuk pertama kalinya. Contoh nyata dari perubahan ini selalu berupa pengemudian otomatis, yang akan membuat jutaan pekerja kehilangan pekerjaan, dimulai dengan 3,5 juta pengemudi truk.

Sebagian dari apa yang membuat prediksi ini kredibel adalah keberhasilan perusahaan mobil dalam mengotomatiskan tindakan tertentu yang terdefinisi dengan baik yang dapat dilakukan mobil. Hal-hal seperti parkir paralel otomatis, perubahan jalur di jalan raya, dan pengereman otomatis. Ini tampak ajaib, dan tampak jelas bahwa kemajuan ini akan terus berlanjut dengan kecepatan yang hampir sama.

Dalam hal ini, pengembangan perangkat lunak, dan khususnya AI, seperti banyak proyek lainnya: lebih mudah di awal, dan bagian akhir – penerapan atau persetujuan akhir – bisa menjadi bagian tersulit. Penting untuk memahami mengapa hal ini berlaku untuk AI, sehingga Anda dapat mengajukan pertanyaan yang bagus tentang aplikasi AI yang Anda temui.

Sebelumnya, kami membahas bagaimana AI hanya dapat menangani kasus yang identik dengan data yang telah ditunjukkan. Dan agar berfungsi, pembelajaran mesin yang mendukung aplikasi AI membutuhkan banyak contoh data ini. Dalam praktiknya, itu berarti Anda akan mendapatkan banyak data yang mudah ditemukan, dan itu akan memberi Anda prototipe yang berfungsi yang dapat melakukan hal-hal menakjubkan. Namun, itu hanyalah prototipe yang berfungsi dalam batasan set data awal yang lebih mudah ditemukan. Inilah sebabnya mengapa demonstrasi tidak menunjukkan produk yang berfungsi, dan mengapa uji coba sangat penting untuk produk yang didukung AI. Dengan menggunakan contoh

mengemudi otomatis, jauh lebih mudah untuk mendapatkan banyak sekali contoh mengemudi di jalan yang kosong, atau mengemudi di jalan bebas hambatan dengan kecepatan 70 mil per jam, dengan sebagian besar mobil hanya melaju lurus.

Karena mereka bisa mendapatkan banyak sekali data tentang kasus penggunaan ini, perusahaan yang mengembangkan mengemudi otomatis dapat menunjukkan prototipe mereka berpindah jalur dengan kecepatan 70 mil per jam di jalan raya normal, atau parkir paralel, dan sebagainya. Namun, kemenangan awal ini tidak terwujud dalam situasi yang lebih rumit seperti masuk ke jalan raya, bergabung, mengemudi di jalan normal – atau apa pun seperti mengemudi di wilayah metropolitan. Dan itu karena situasi ini terjadi dalam banyak cara, dan ada lebih banyak lagi kasus ekstrem yang kami perkenalkan di Bab 2.

Ada dua poin besar yang muncul dari perbedaan antara keberhasilan awal aplikasi yang sempit, dan penundaan yang lama pada aplikasi yang lebih luas. Ini adalah tingkat akurasi yang dibutuhkan dan meningkatnya kesulitan dalam mendapatkan data pada kasus ekstrem. Mari kita bahas lebih dalam mengenai isu-isu ini, karena isu-isu ini akan menentukan apa yang dapat dan tidak dapat Anda capai di lokasi kerja dengan AI.

7.1 TINGKAT AKURASI

Memahami seberapa akurat AI yang Anda butuhkan memiliki konsekuensi besar terhadap jumlah data yang Anda butuhkan, karena jumlah data yang Anda butuhkan akan menjadi sangat, sangat tinggi seiring dengan meningkatnya akurasi Anda. Prinsip Pareto, yang dikenal sebagai aturan 80/20, berlaku di sini, tetapi jauh lebih buruk. Prinsip ini lebih seperti 99% data Anda akan digunakan untuk mendapatkan 1% terakhir, sehingga 1% terakhir terlalu mahal untuk sebagian besar aplikasi. Jika harus 99,9% akurat, pendekatan AI saat ini masih belum layak untuk sebagian besar situasi dunia nyata.

Misalnya, Amazon menggunakan AI untuk memberi Anda rekomendasi tentang apa yang mungkin ingin Anda beli. Menurut pengalaman saya, rekomendasi ini tidak terlalu akurat, meskipun saya membeli banyak buku di situsnya. Namun, itu tidak terlalu penting, tidak ada yang dirugikan oleh akurasi rekomendasi yang rendah, dan saya tetap menemukan apa yang saya cari. Kita dapat mengatakan bahwa tingkat akurasinya tidak perlu terlalu tinggi agar dapat diterima. Amazon tidak mempublikasikan keakuratan rekomendasi mereka, tetapi mari kita duga keakuratannya adalah 75%. Yang menarik adalah Amazon memiliki jumlah data yang sangat menakjubkan, dan mereka adalah penyedia komputasi awan terbesar di dunia, jadi mereka jelas tidak kekurangan sumber daya untuk mengolah data tersebut. Masalahnya di sini adalah bahwa masalahnya sendiri terlalu luas, dan selera setiap orang cukup berbeda sehingga mendapatkan data yang cukup untuk menjadi lebih akurat dari ini menjadi lebih sulit seiring dengan meningkatnya tingkat keakuratan yang Anda inginkan. Namun mungkin Amazon tidak berusaha keras untuk membuat keakuratannya jauh lebih tinggi, dengan melakukan survei atau aktivitas lain yang akan mempermudah masalah tersebut. Mereka berada pada tingkat keakuratan yang tepat untuk apa yang ingin mereka capai – terkadang mendapatkan rekomendasi yang "salah" berarti Anda mencoba sesuatu yang tidak terduga, yang merupakan hal yang baik untuk situs e-commerce.

Sebaliknya, FaceID Apple sebenarnya adalah fitur keamanan, jadi tingkat keakuratan merupakan masalah besar. Apple memang mempublikasikan hal ini, dan mereka memperkirakan kemungkinan FaceID salah adalah sekitar satu dari sejuta, atau keakuratannya adalah 99,9999%. Itu adalah standar yang tinggi, dan untuk mencapainya Apple menggunakan lebih dari sekadar kamera, mereka menggunakan laser untuk memindai lebih dari 30.000 titik di wajah Anda. Apple membuat bagian AI dari masalah tersebut "semudah" mungkin dengan mengumpulkan begitu banyak titik referensi wajah, dan melakukannya dengan pemindai laser, bukan hanya gambar. Anda yang telah melakukan perekaman realitas di lokasi kerja akan segera memahami seberapa akuratnya hal itu.

Jadi kita punya dua contoh, kasus rekomendasi e-commerce di mana masalahnya luas, tetapi akurasi yang dibutuhkan rendah, jadi mereka tidak mencoba membuat masalahnya lebih mudah. Contoh yang kontras adalah FaceID, di mana masalahnya sengaja dipersempit, akurasi yang dibutuhkan tinggi, dan aplikasinya dibuat untuk menangkap banyak informasi yang memberikan akurasi tinggi.

Tidak sulit untuk memperluas pemikiran itu ke lokasi kerja konstruksi – berapa banyak masalah yang bisa dipersempit dan mengumpulkan banyak data, dan berapa banyak masalah yang selalu luas dan karenanya menjadi masalah yang sulit? Kembali ke kendaraan otonom, sekali lagi Anda melihat bahwa ketika tugas spesifik yang harus diselesaikan relatif sempit dan kita dapat menangkap banyak sekali data, AI bekerja dengan baik. Parkir, pengereman darurat, dan perubahan jalur – semua ini sangat berharga, dan untuk dua yang pertama, manusia sering kali tidak begitu ahli dalam hal tersebut. Namun, hal-hal tersebut merupakan bagian kecil dari keseluruhan berkendara.

Mengemudi di jalan yang ramai tidak hanya lebih rumit, tetapi juga sarat dengan potensi kejadian tak terduga yang mungkin memerlukan reaksi. Dan itu membawa kita ke poin utama kedua, yaitu perbedaan antara peningkatan kinerja dan kuantitas data yang diperlukan untuk mencapai kinerja tersebut.

Menangkap Batas

Kebanyakan hal yang kita lakukan sering kali membutuhkan kinerja yang hampir sama setiap saat. Mengemudi ke tempat kerja setiap hari dapat terasa seperti lagu yang diputar secara otomatis. Anda keluar dari jalan masuk, berkendara di jalan, berhenti di beberapa rambu berhenti, masuk ke jalan raya, keluar dari jalan raya, beberapa rambu berhenti lagi dan Anda berada di tempat parkir.

Kecuali sesekali, orang bodoh di depan Anda mengerem tanpa alasan. Atau bola sepak keluar dari halaman di sebelah kiri Anda. Seseorang menerobos rambu berhenti, atau rambu berhenti tersembunyi di balik semak-semak, atau ada kabut, atau ada mobil yang diparkir secara ilegal, dan seterusnya.

Daftar hal-hal yang terjadi sepanjang waktu biasanya pendek, dan karena terjadi sepanjang waktu, mudah untuk mendapatkan data yang mewakili semua kejadian yang sering terjadi ini. Ini berarti membangun sistem AI yang dapat menangani semua kejadian yang sering terjadi ini dapat dikelola, dan telah dilakukan cukup awal dalam proses pengembangan kendaraan otonom. Misalnya, Waymo, unit Google yang mengembangkan kendaraan otonom

dan dianggap sebagai salah satu yang paling maju dalam industri ini, telah menempuh jarak sekitar 5 juta mil pada tahun 2018. Selama kurun waktu tersebut, akurasi mereka terus membaik, sebagaimana diukur dari persentase waktu pengemudi harus mengambil alih kendali dari AI. Hal ini dikenal sebagai tingkat pelepasan kendali, dan terus menurun.

Masalah bagi AI secara umum, dan mobil secara khusus, adalah bahwa hal-hal "lain" yang terjadi, kasus-kasus ekstrem seperti pengereman acak, menerobos rambu berhenti, dan sebagainya, tidak sering terjadi, dan hampir menurut definisi jumlahnya lebih banyak daripada hal-hal yang sering terjadi. Jadi, Anda memiliki daftar kejadian yang jauh lebih banyak, jauh lebih sedikit. Pikirkan berapa kali dalam hidup Anda, Anda benar-benar melihat seseorang menerobos lampu merah. Mungkin 10, 15 kali dalam seluruh hidup Anda. Itu mungkin 15 kali terlalu banyak, tetapi itu sangat kecil dibandingkan dengan ribuan kali Anda berhenti di lampu merah dan tidak ada yang menerobosnya. Hasil praktisnya adalah bahwa awal pembuatan proyek AI akan tampak hebat – demo akan berhasil. Massa akan bersorak, jurnalis akan menulis tentang masa depan yang dramatis dengan manfaat bagi konsumen dan kematian pekerja. Kemudian beberapa bulan atau tahun berlalu dan kemajuan tampaknya melambat, lalu hampir terhenti. Pada saat yang sama, perusahaan yang mengembangkan AI akan memberi tahu Anda bahwa mereka telah menemukan cara untuk menghasilkan data dalam jumlah besar, dan akurasinya tetap tidak membaik secepat itu.

Hal ini karena saat Anda semakin mendekati akurasi 100%, daftar kasus tepi yang potensial bertambah secara eksponensial, tetapi jumlah sebenarnya kasus tepi ini akan terjadi tetap kecil, jadi Anda perlu melakukannya dalam jumlah waktu yang terus bertambah secara eksponensial. Ada dua kemungkinan respons terhadap situasi ini: Dapatkan lebih banyak data atau buat masalahnya lebih kecil.

Kedua pendekatan tersebut sedang dicoba pada kendaraan otonom. Di AS, Waymo terus memiliki ribuan kendaraan di jalan. Dengan sumber daya AI Google, mereka mungkin akan memecahkannya suatu hari nanti, hanya saja tidak secepat yang diharapkan. Demikian pula, setiap Tesla di jalan raya terus-menerus menangkap data, yang jumlahnya bisa mencapai jutaan mil data, jadi mereka mungkin dapat memecahkan masalah tersebut lebih cepat.

Seperti yang telah kita lihat, pendekatan ini berarti bahwa teknologi kendaraan otonom akan menjadi proses pengenalan fitur per fitur yang lambat. Lebih seperti teknologi bantuan pengemudi dalam waktu dekat.

Sebaliknya, rute bus, antar-jemput bandara, dan aplikasi angkutan umum lainnya adalah cara yang bagus untuk mempersempit masalah. Anda dapat membatasi lajurnya, membatasinya pada halte yang telah ditentukan sebelumnya, dan seterusnya. Mungkin begitulah cara kendaraan otonom benar-benar menjadi hal yang umum selama beberapa tahun ke depan.

Untuk memahami betapa menakjubkannya otak manusia, pertimbangkan ini: Anda mengemudi sepanjang waktu, hampir tidak memerlukan perhatian Anda. Namun, tugas mudah ini telah membingungkan semua solusi AI yang telah dicoba. Perusahaan dengan pendanaan terbaik di dunia baru saja mendapatkan demo mereka untuk melakukan beberapa hal di dunia nyata.

Pikirkan tentang itu saat Anda membaca laporan yang membuat Anda terengah-engah tentang pengangguran di masa depan karena AI.

7.2 PELAJARAN TENTANG AKURASI AI

Semua ini tidak berarti bahwa AI tidak penting, atau tidak akan membantu Anda dalam pekerjaan Anda di tahun-tahun mendatang. Namun, kita perlu bertanya apa yang kita minta AI lakukan, dan seberapa baik masalah tersebut didefinisikan. Mari kita uraikan menjadi tiga pertanyaan yang dapat Anda ajukan tentang aplikasi apa pun, dan apakah AI akan cukup mampu membantu aplikasi tersebut:

1. Seberapa akurat aplikasi ini membutuhkan AI?
2. Seberapa sempit kita dapat mendefinisikan aplikasi ini dan membuatnya memberikan nilai tambah?
3. Berapa banyak data yang dapat kita kumpulkan, dan seberapa mahal biaya pengumpulannya?

Meningkatkan Pekerjaan Manusia

Pertanyaan tentang tingkat akurasi yang diperlukan bermuara pada pemahaman apakah aplikasi tersebut berinteraksi langsung dengan dunia, atau merupakan sesuatu yang dapat melibatkan "manusia."

Manusia terlibat dalam produk AI lebih dari yang umumnya dilaporkan. Banyak aplikasi AI yang berhadapan dengan pengguna yang akan Anda lihat melibatkan manusia dalam beberapa hal, baik itu salah satu aplikasi pengaturan janji temu AI, pengenalan foto, atau yang lainnya. Ada seseorang, sering kali pekerja kontrak di luar negeri, yang memeriksa pekerjaan AI untuk memastikan tidak ada kesalahan. Tidak ada yang salah dengan ini, jika berhasil dan tidak membuat aplikasi terlalu mahal untuk diskalakan. Apakah manusia diperlukan agar aplikasi berfungsi adalah pertanyaan yang bagus untuk diajukan jika perusahaan rintisan mendatangi Anda dengan beberapa teknologi AI baru – banyak perusahaan tahap awal menggunakan manusia untuk menebus fakta bahwa mereka belum melatih sistem mereka dengan cukup data.

Melibatkan manusia berarti bahwa semua pengenalan gambar, penangkapan realitas, dan aplikasi transkripsi ucapan AI akan memiliki tingkat akurasi AI yang lebih rendah, karena Anda akan memiliki lapisan akal sehat ekstra untuk memastikan kesalahan tidak muncul dalam proses. Model tersebut tidak harus melakukan hal-hal yang sangat sulit, seperti menangani kasus-kasus khusus. Model tersebut hanya mengotomatiskan langkah-langkah umum yang sering diulang.

Sebaliknya, jika perangkat lunak secara langsung mengeluarkan output-nya ke dunia, baik dengan menjalankan mesin atau mengirimkan informasi atau analisis secara langsung ke tim, pelanggan, atau manajemen Anda, tidak ada manusia yang dapat menemukan kesalahan. Jadi, Anda perlu mengajukan pertanyaan berikutnya: apa risikonya jika AI salah? Tiga contoh menggambarkan maksud saya:

1. ***Menavigasi drone vs. backhoe***. Drone berada di udara, jadi berada di lingkungan yang jauh lebih sederhana daripada mesin di darat. Drone hampir tidak akan pernah

terhalang oleh apa pun yang tidak dapat mereka lihat sebelumnya, dan mereka dapat bergerak ke hampir semua arah untuk menghindari sesuatu yang tidak terduga. Mesin di darat, seperti backhoe, dikelilingi oleh orang, material, dan mesin lain yang melakukan hal-hal yang tidak terduga, yang dapat membahayakan. Tingkat akurasi yang dibutuhkan untuk kedua aplikasi ini berbeda – drone dapat memiliki tingkat akurasi yang lebih rendah, sedangkan backhoe akan jauh lebih tinggi.

2. ***Analisis data vs. visualisasi data***. Ada aplikasi AI yang dapat menjalankan analisis secara otomatis untuk Anda. Namun tanpa manusia untuk memeriksa kesimpulan mereka, adalah ide yang buruk bagi sistem AI untuk langsung menerbitkan laporan, dan saya rasa tidak ada yang melakukannya sekarang. Sebaliknya, meminta AI untuk memvisualisasikan data, atau memilih skema warna tidak menciptakan risiko nyata. Bahkan di sana Anda mungkin tidak ingin "melepaskannya," tetapi taruhannya jelas lebih rendah antara menerbitkan kesimpulan dan memilih cara memvisualisasikannya.
3. ***Model yang dibangun vs. penangkapan realitas.*** Mirip dengan contoh #2, model yang dibangun akan dilihat oleh sejumlah orang penting, sehingga akurasinya bisa sangat penting. Model tersebut perlu diperiksa oleh manusia untuk memastikan akurasinya. Sebaliknya, penangkapan realitas otomatis yang semakin umum adalah hal yang tepat untuk ditangani oleh AI, karena model tersebut merupakan masukan ke dalam proses desain dan penyempurnaan model lain yang dijalankan oleh orang-orang yang dapat mengelolanya jika ada beberapa kesalahan.

Panduan Praktis untuk Keterbatasan

Kita telah menemukan gagasan keterbatasan ini beberapa kali dalam buku ini. Perangkat lunak bagus dalam tugas-tugas yang sempit dan terspesialisasi yang tidak dapat dilakukan oleh manusia, dan manusia bagus dalam masalah-masalah yang luas dan masuk akal yang tidak dapat dilakukan oleh perangkat lunak. Apa yang kita maksud dengan "keterbatasan?" Dalam hal mendefinisikan masalah AI, yang dimaksud dengan sempit bukanlah masalah itu sendiri, melainkan data yang dapat Anda kumpulkan di dunia nyata, dan itu bergantung pada berapa banyak cara masalah itu muncul, berapa banyak cara untuk menyelesaikan berbagai versi masalah ini, dan berapa banyak elemen yang dapat menjadi bagian dari penyelesaian masalah. Anggap saja sebagai tiga daftar yang dikalikan bersama untuk membuat daftar lengkap tentang berapa banyak jenis data yang Anda perlukan:

1. Berapa banyak cara masalah itu muncul?
2. Berapa banyak cara Anda dapat menyelesaikan setiap versi?
3. Berapa banyak faktor yang dapat terlibat dalam solusi di dunia nyata?

Misalnya, dalam contoh kendaraan otonom (AV) kita lagi, kita memiliki sejumlah jenis tugas mengemudi yang wajar, katakanlah 50. Dan kita memiliki sejumlah cara yang wajar untuk menyelesaikan setiap masalah, katakanlah rata-rata 10. Secara keseluruhan, itu hanya 500 kombinasi masalah/solusi yang unik. Jadi mungkin ada 5 cara untuk keluar dari jalan masuk, tetapi ada 15 cara untuk melewati tempat parkir.

Daftar ketiga adalah bagian yang sulit, karena jumlahnya hampir tak terbatas. Dan karena Anda tidak dapat mengabaikan contoh yang mungkin, daftar kombinasi Anda menjadi sangat panjang.

Jadi, kita punya tiga daftar, dua masuk akal dan yang ketiga tidak. Daftar ketiga itu tentang seberapa banyak kompleksitas dunia yang Anda biarkan masuk ke aplikasi Anda. Mobil di luar sana tidak dapat mengendalikannya, jadi dunia yang luas dan luas dengan segala kemegahannya adalah bagian dari masalah mobil otonom, kecuali Anda membatasinya dengan rute yang ditetapkan, jalur AV khusus, dan strategi lain yang telah kita bahas sebelumnya.

Itulah pelajaran pertama dari setiap aplikasi AI: bagaimana Anda dapat membatasi paparan terhadap variasi dunia nyata yang tak terbatas?

Integritas Data: Sisi Lain dari Sempit

Sepenting apa pun untuk mempersempit pertanyaan atau tugas Anda ke pertanyaan atau tugas yang dapat ditangani oleh model AI, penting juga untuk merancang proses Anda sehingga Anda dapat mengumpulkan cukup data untuk melatih model dengan baik. Ini bukan hanya tentang kuantitas, tetapi juga akan sangat bergantung pada rentang contoh yang Anda sertakan – dan biasanya memerlukan sedikit kreativitas di pihak perancang proses.

Karena AI akan belajar dari data, dan hanya dari data, Anda harus berhati-hati tentang AI yang mempelajari hal-hal yang tidak relevan, dan dibutuhkan sedikit imajinasi untuk memikirkan hal-hal tersebut. Misalnya, jika Anda melatih sistem Anda untuk mengenali kunci inggris, dan setiap kunci inggris berukuran 1/8 inci berada di atas meja kayu, tetapi setiap kunci inggris berukuran 1/16 inci berada di atas beton, Anda dapat bertaruh bahwa sistem akan berpikir bahwa setiap kunci inggris di atas beton berukuran 1/16 inci, dan setiap kunci inggris di atas kayu berukuran 1/8 inci.

Imajinasi akan membawa Anda cukup jauh dalam kasus ini, tetapi Anda juga perlu menguji sistem Anda dan melihat seberapa baik kinerjanya – dan di sini sekali lagi data yang Anda uji harus seluas mungkin.

Integritas data lebih dari sekadar akurasi untuk alasan fungsional. Hal ini memiliki implikasi etika yang besar, karena kumpulan data yang tidak mewakili populasi yang akan diterapkan akan menyebabkan bias dalam model yang dibuatnya. Bias AI telah disebut sebagai masalah untuk sistem AI yang digunakan untuk kepolisian, aplikasi pinjaman, dan khususnya pemrosesan bahasa alami. Yang terakhir ini menarik karena pengembang perangkat lunak sering menggunakan kumpulan data besar bahasa nyata untuk melatih model mereka, dan bahasa nyata sering kali mencerminkan ide dan pendapat yang sebenarnya tidak relevan dengan apa yang Anda coba latih, tetapi tetap muncul. Masalah praktis untuk aplikasi konstruksi AI adalah bahwa bias akan berarti keluaran yang buruk. Ini berarti bahwa sistem tidak dapat diandalkan dan karenanya kurang bermanfaat. Orang cenderung mempercayai apa yang tertulis atau apa yang muncul dari sistem perangkat lunak yang mahal, terutama yang berasal dari perusahaan besar atau dari "laboratorium inovasi". Hasilnya, tidak sulit untuk membayangkan bias yang tidak Anda sadari telah muncul setahun kemudian, jauh

setelah sistem menjadi "normal", dan bahwa hasil bias ini tidak dipertanyakan karena tidak ada yang memeriksanya terhadap kenyataan, atau bahkan intuisi mereka sendiri.

Ingatlah bahwa AI sedikit bodoh. Ia membutuhkan bantuan Anda.

7.3 PENGUMPULAN DATA

Sebagian besar pembahasan sebelumnya mengasumsikan bahwa suatu organisasi mampu mengumpulkan data yang mereka butuhkan, dalam jumlah yang dibutuhkan oleh proses pengembangan AI yang sebenarnya.

Berapa banyak data?

Asumsikan bahwa model asli akan membutuhkan ratusan ribu, hingga jutaan contoh, tergantung pada seberapa sempit Anda telah membuat masalah tersebut. Untuk model yang telah dilatih sebelumnya, Anda sering kali dapat melakukannya dengan jumlah yang jauh lebih sedikit, tetapi sebagian besar aplikasi AI tidak memiliki model yang telah dilatih sebelumnya. Sering dikatakan bahwa setiap proyek pembelajaran mesin 95% tentang data. Ini karena pengumpulan data biasanya memerlukan kecerdikan, tetapi juga karena data yang ada hampir tidak pernah dalam format yang tepat, memiliki data yang buruk, atau tidak lengkap. Menyempurnakan data, mengumpulkan kembali, dan membuatnya berfungsi dalam sistem pelatihan adalah pekerjaan keras yang membutuhkan waktu khusus dari seorang ilmuwan data. Jadi, sebelum membeli produk AI baru yang mengilap itu, tanyakan kepada tim yang mengembangkannya berapa banyak data yang akan mereka butuhkan, seberapa luas kumpulan data yang akan dibutuhkan, dan bagaimana mereka akan menguji bias baik dalam jangka pendek maupun jangka panjang.

Pembelajaran Transfer: AI yang Mudah

Sebagian besar perusahaan tidak memiliki cukup data untuk melatih sistem AI. Untungnya, kumpulan data yang besar terkadang tidak diperlukan, karena ada model yang telah dilatih dan dapat melakukan banyak hal yang Anda inginkan.

Google melatih model pemrosesan bahasa alami mereka, BERT, pada semua konten Wikipedia. Mereka telah menyediakan model itu untuk digunakan oleh perusahaan, dan dengan upaya yang relatif rendah Anda dapat membuat BERT berfungsi untuk tugas pemrosesan bahasa alami Anda – seperti mungkin mengenali instruksi, kata, dan frasa tertentu mengenai perdagangan MEP. BERT membuat chatbot, atau agen suara, jauh lebih mampu dan "alami," dan bekerja dengan sangat sedikit data pelatihan – kami berhasil membuatnya berfungsi dengan skrip dua halaman, misalnya. Ini disebut pembelajaran transfer, dan berfungsi baik untuk pengenalan bahasa dan gambar, karena ada banyak model berkualitas tinggi yang dapat diakses. Untuk penggunaan potensial lainnya, pertama-tama lakukan pencarian Google tentang masalah tersebut, karena siapa pun yang memiliki model yang dapat digunakan seperti ini hampir pasti telah menerbitkannya, dan mungkin memiliki beberapa studi kasus. Jika itu menunjukkan beberapa hasil, inilah saatnya untuk melibatkan ilmuwan data, dan jika pencarian tidak segera menunjukkan hasil, sekali lagi libatkan ilmuwan

data, karena mereka mungkin mengetahui contoh yang akan berhasil meskipun tidak cocok satu lawan satu.

Namun, ada peringatan. Bahkan model yang telah dilatih sebelumnya ini memerlukan ilmuwan data dan programmer agar dapat berfungsi. Masalah yang Anda pecahkan adalah kebutuhan akan banyak data, dan mereka melakukannya dengan baik. Selanjutnya kita akan membahas pertanyaan tentang bagaimana Anda dapat membuat, atau membeli, aplikasi AI. Aplikasi AI

Anda dapat menerapkan kecerdasan buatan pada alur kerja Anda dengan tiga cara utama:

- membangunnya dari awal
- membangun aplikasi di sekitar API kecerdasan buatan milik orang lain
- membeli produk atau layanan yang telah melakukan semua ini untuk Anda.

Sebagian besar perusahaan akan melakukan opsi kedua dan ketiga ini, tetapi mari kita bahas ketiganya.

7.4 MEMBANGUN AI ANDA SENDIRI

Sebelumnya kami memperkenalkan banyak hal yang diperlukan untuk membangun model Anda sendiri, dan banyak hal yang perlu Anda perhatikan, jadi mari kita uraikan secara lebih rinci seperti apa proses itu dalam kehidupan nyata.

1. *Identifikasi kebutuhan:* Anda hanya akan mempertimbangkan AI jika Anda memiliki sesuatu yang ingin Anda selesaikan, beberapa hal perlu ditangani. AI cocok jika Anda ingin mengotomatiskan sesuatu, baik itu mengenali elemen umum, memandu pengguna melalui perangkat lunak, mengenali masalah keselamatan di lokasi kerja, dan fungsi lainnya seperti ini.
2. *Tentukan tingkat akurasi:* kebutuhan dan aplikasi di dunia nyata akan mendorong hal ini. Ikuti langkah-langkah di bagian sebelumnya untuk merekayasa masalah sehingga tingkat akurasi yang Anda perlukan serendah mungkin.
3. *Tetapkan cakupannya:* tidak semuanya akan menjadi "AI", beberapa hanya akan menjadi perangkat lunak biasa. Cobalah untuk membuat bagian AI sesempit dan sefokus mungkin. Langkah ini harus mencakup desain perangkat lunak lainnya yang akan menghadirkan AI Anda, serta proses yang akan memasukkan data kembali ke AI sehingga terus belajar.
4. *Tetapkan data:* jenis data apa yang diperlukan untuk melatih AI? Bagaimana Anda akan mengumpulkannya? Siapa yang akan mengawasi ini – haruslah seorang ilmuwan data profesional.
5. *Tentukan modelnya*: sering kali ada banyak cara yang mungkin berhasil, dan ada ratusan model yang tersedia secara gratis. Ini jelas merupakan pekerjaan seorang ilmuwan data, dan sangat penting bahwa orang ini memiliki pengalaman yang solid dan mendalam yang telah Anda verifikasi dengan mitra sebelumnya, karena Anda tidak mungkin dapat memeriksa pekerjaan mereka secara langsung.
6. *Kumpulkan data:* Terkadang Anda sudah memiliki banyak data, terkadang Anda perlu mengumpulkannya. Dalam banyak kasus, kreativitas diperlukan untuk mendapatkan

data yang Anda inginkan. Beberapa perusahaan telah membuat seluruh bisnis sampingan yang didedikasikan untuk mengumpulkan data untuk tujuan utamanya – Anda mungkin memutuskan untuk melakukan sesuatu seperti membuat aplikasi atau proses lain untuk mempercepat pengumpulan data.

7. *Latih model:* Ini akan melibatkan banyak perhitungan angka, penyetelan, dan pengujian internal. Cara standar untuk melatih model adalah dengan menggunakan sekitar 70% data Anda untuk melatih model, lalu 30% untuk mengujinya. Hal itu berfungsi untuk memastikan Anda telah melatih model terhadap data yang dapat Anda kumpulkan, tetapi tidak melindungi Anda dari data yang tidak lengkap atau bias.
8. *Uji model di dunia nyata:* coba model dengan tim Anda sendiri, lalu dengan pelanggan. Di sinilah kasus-kasus ekstrem dan data yang tidak Anda antisipasi akan terungkap, dan dapat mengharuskan Anda untuk melalui langkah 5 dan 6 lagi.
9. *Rancang proses data dan pelatihan yang berkelanjutan:* Keajaiban pembelajaran mesin adalah ia dapat terus belajar dari waktu ke waktu, dan dalam prosesnya meningkat. Yang penting adalah bahwa data dunia nyata, dan hasilnya, disertakan. Jadi jika AI membuat klasifikasi yang buruk, Anda perlu memasukkannya dalam set pelatihan yang sedang berlangsung. Dan proses ini perlu memiliki perlindungan anti-bias yang setidaknya seketat pengumpulan data asli.
10. *Finalisasikan produk dan kirim:* sekarang Anda akan memiliki beberapa pemahaman tentang seberapa sering model bekerja dan seberapa sering ia tersandung. Dengan menggunakan informasi ini, Anda dapat memutuskan apakah, dan bagaimana, manusia akan dilibatkan. Manusia dalam proses ini mungkin tidak permanen, karena sistem akan membaik seiring penggunaan.

Perlu dicatat bahwa Anda perlu melakukan proses yang sama ini baik jika Anda mengembangkan model dari awal, atau jika Anda mengambil salah satu model yang "dapat dilatih", seperti BERT. Proses ini jauh lebih cepat dengan model yang telah dilatih sebelumnya.

7.5 MEMBANGUN DENGAN API

Google, AWS, IBM, Microsoft, dan perusahaan cloud lainnya menyediakan rangkaian lengkap fungsi kecerdasan buatan yang dapat Anda akses melalui API. API ini biasanya sudah matang, dilatih pada data dalam jumlah besar, dan yang terpenting, terus-menerus disetel dan diperbarui untuk performa. Bagian terakhir itu saja, bahwa API disetel untuk performa, sudah cukup menjadi alasan untuk menggunakan API ini, karena model AI yang Anda bangun sendiri bisa sangat besar, haus daya, dan sulit untuk dijalankan sesuai keinginan Anda.

API dari perusahaan perangkat lunak besar akan disertai dengan dokumentasi yang jelas, beberapa tingkat dukungan, dan kemungkinan besar komunitas yang pernah bekerja sama dengan mereka, sehingga Anda dapat "mencarinya di Google" saat Anda mengalami kesulitan. Sampai batas tertentu, hal ini juga berlaku untuk perusahaan lain; misalnya, Houndify, perusahaan di balik aplikasi pengenalan musik

SoundHound, menawarkan kemampuan pemrosesan bahasa alami yang luar biasa dan tim hubungan pengembang yang profesional. Meskipun spesifikasi tentang apa yang akan

Anda bangun, dan bagaimana Anda akan membangunnya akan bergantung pada bisnis dan tim Anda sendiri, ada beberapa hal yang perlu Anda ketahui:

1. Biasanya lebih baik menggunakan API daripada membangun sendiri.
2. Cobalah untuk melakukan outsourcing sesedikit mungkin, dan batasi outsourcing ke luar negeri. Tim yang melakukan ini harus berkomunikasi secara terus-menerus dan jelas dengan pemilik bisnis, dan itu selalu lebih baik jika itu adalah tim Anda.
3. Petakan seberapa sering Anda akan menggunakan API. Pengembang Anda harus menjadi bagian dari ini sehingga Anda memahami seberapa banyak Anda melakukan hal-hal yang dapat dikenai biaya. Jangan berasumsi karena biaya API sangat rendah maka dalam skala besar biayanya akan rendah. Biaya tersebut dapat bertambah, seperti halnya layanan lain seperti penyimpanan, email, dan lainnya yang biasanya diperlukan.
4. Jangan abaikan UX. Sertakan pengujian pengguna secara eksplisit dalam penjadwalan Anda, bukan hanya untuk melihat apakah itu berhasil, tetapi untuk melihat apakah orang biasa yang tidak terhubung dengan proyek tahu cara membuatnya berhasil.

Anda dapat dan harus membuka platform cloud Google, konsol AWS Machine Learning/AI Services, Microsoft Azure Cognitive Services, IBM Watson, dan lainnya, dan menjelajah sebentar. Ini penting karena masing-masing perusahaan ini telah membangun berbagai layanan yang dapat Anda gunakan, lebih dari sekadar pengenalan gambar generik atau pembuat chatbot. Misalnya, berikut ini adalah apa yang akan Anda lihat di halaman Microsoft Azure Cognitive Services, tempat mereka memiliki beberapa opsi tingkat tinggi, termasuk keputusan, bahasa, ucapan, penglihatan, dan pencarian web. Tab "bahasa" meliputi:

- Mengekstrak makna dari teks yang tidak terstruktur
- PRATINJAU Pembaca Imersif
  Membantu pembaca dari semua kemampuan memahami teks menggunakan isyarat audio dan visual
- Pemahaman Bahasa
  Membangun pemahaman bahasa alami ke dalam aplikasi, bot, dan perangkat IoT
- QnA Maker
  Buat lapisan tanya jawab percakapan pada data Anda
- Analisis Teks
  Deteksi sentimen, frasa kunci, dan entitas bernama
- Teks Penerjemah
  Deteksi dan terjemahkan lebih dari 60 bahasa yang didukung.

Hanya dalam daftar ini mereka memiliki cara untuk membantu Anda membuat Tanya Jawab, mendeteksi sentimen, dan melakukan penerjemahan. Demikian pula, di bawah tab "pencarian" mereka, mereka memiliki sekitar 10 fungsi mirip pencarian yang dapat Anda tambahkan, seperti pemeriksa ejaan, pencarian video, dan banyak lagi.

Di bawah bagian "Ucapan", mereka memiliki API yang akan mentranskripsikan audio, termasuk saat itu menjadi bagian dari video, menerjemahkan, dan mendeteksi siapa yang berbicara. Jadi bayangkan jika Anda melakukan rapat Zoom dan Anda ingin membuat transkrip

otomatis – dua dari API ini bersama-sama berarti Anda bisa mendapatkannya hampir secara real time, dengan pembicara yang teridentifikasi. Saya belum membuat aplikasi itu tetapi mereka membuatnya seringan mungkin. Platform cloud Google sering kali menjadi salah satu yang terbaik dalam hal performa, meskipun UI mereka terkenal sulit. Sama seperti Microsoft, menghabiskan waktu di Google akan menunjukkan kepada Anda berbagai kemungkinan, mulai dari visi mesin hingga ucapan, teks, hingga pengklasifikasi umum. Google umumnya lebih diarahkan kepada pengembang, jadi berikan diri Anda sedikit lebih banyak waktu untuk mencoba-coba, karena saat Anda siap untuk mulai membangun sebagai platform, mereka memiliki perangkat alat yang paling lengkap.

Saat berhadapan dengan salah satu platform ini, pastikan pengembang perangkat lunak Anda benar-benar melihat-lihat semua platform cloud, karena mereka biasanya akan mencoba untuk menempatkan semuanya di penyedia cloud yang paling mereka kenal, yang belum tentu merupakan pilihan yang tepat. Dorong mereka untuk menghubungkan API dari satu penyedia ke penyedia lainnya jika itu adalah solusi yang tepat.

7.6 MEMBELI AI ANDA

Bagi sebagian besar perusahaan, dalam sebagian besar kasus penggunaan, Anda tidak akan membangun AI Anda sendiri dari awal, atau membangun produk di sekitar API. Sebaliknya, Anda akan menggunakan kecerdasan buatan dalam produk yang sudah jadi, baik dari perusahaan besar seperti Autodesk atau Procore, atau perusahaan rintisan yang lebih kecil seperti Smartvid.io atau openspace.ai. Masing-masing perusahaan ini akan memiliki pendekatan yang sangat berbeda, didorong oleh latar belakang bisnis mereka dan juga teknologi itu sendiri.

Percontohan sangat penting di sini, untuk memastikan AI benar-benar melakukan apa yang mereka katakan, dan untuk memahami keamanan data dalam praktiknya. Jika mereka menggunakan manusia untuk mengawasi kinerja produk mereka, mereka perlu memberi tahu Anda dan memastikan bahwa tidak akan ada pelanggaran data. Tidak ada jaminan bahwa penyedia AI Anda menggunakan manusia untuk kontrol kualitas, tetapi Anda harus bertanya.

Berurusan dengan Perusahaan Rintisan

Siapa pun yang pernah bekerja dengan perusahaan rintisan akan setuju bahwa perusahaan rintisan sangat berbeda dengan perusahaan besar. Perusahaan rintisan biasanya belum menyusun model dukungan pelanggan, sehingga bisa jadi berlebihan atau kekurangan sumber daya, biasanya yang terakhir. Perusahaan rintisan biasanya mencari tahu campuran produk dan pendekatan pasar secara keseluruhan, jadi mereka mungkin tidak langsung memiliki semua yang Anda inginkan, tetapi mereka sering kali akan melakukan beberapa penyesuaian untuk Anda.

Perusahaan rintisan cenderung menjadi sumber ide yang benar-benar baru. Para pendiri perusahaan rintisan akan sepenuhnya berfokus pada serangkaian fitur dan fungsi pengguna akhir yang sempit, jadi mereka akan bekerja sama secara mendalam dengan Anda untuk memecahkan masalah Anda. Sebagai mantan dan kemungkinan pendiri perusahaan rintisan di masa mendatang, saya dapat mengatakan bahwa Anda memang mendapatkan

perhatian dari pengambil keputusan akhir, dan sangat sedikit ketidakfleksibelan yang sulit dihindari oleh perusahaan besar. Manfaat dari perusahaan rintisan, tetapi juga dukungan ekstra yang mereka butuhkan, adalah alasan mengapa banyak kontraktor umum telah membentuk tim inovasi, sehingga sisi yang lebih kasar dari perusahaan rintisan perangkat lunak dapat diintegrasikan ke dalam operasi yang lebih besar dari perusahaan kontraktor.

Terlepas dari kenyataan berurusan dengan perusahaan yang lebih kecil dan baru muncul, penting untuk dipahami juga bahwa sangat sedikit perusahaan rintisan yang memiliki akses ke sejumlah besar data, yang berarti bahwa sebagian besar dari mereka akan membuat model mereka menggunakan pustaka sumber terbuka, atau beberapa API yang kami jelaskan sebelumnya. Ini seharusnya tidak menjadi masalah, karena Anda benar-benar membeli perangkat lunak yang melakukan hal-hal yang bermanfaat, bukan "AI" itu sendiri. Namun, itu berarti bahwa apa yang mungkin tampak seperti solusi yang benar-benar unik mungkin tidak, dan jika Anda menyukai solusi perusahaan tertentu, Anda harus melihat-lihat dan melihat siapa lagi yang mencoba hal yang sama. Nilai bagi Anda sebagai pengguna dan pembeli adalah bahwa, bahkan jika Anda menyukai perusahaan yang ada, Anda sering dapat belajar dari situs web, studi kasus, dan "kertas putih" yang tak terelakkan yang dihasilkan oleh perusahaan rintisan lainnya.

Akurasi AI

Sebagian besar aplikasi kecerdasan buatan melaporkan hasil seolah-olah hasil tersebut dipilih dengan keyakinan penuh. Namun, penting untuk diingat bahwa fitur kecerdasan buatan merupakan hasil matematika yang canggih, matematika yang menggunakan pelatihan sebelumnya untuk memperkirakan kemungkinan bahwa pola, gambar, atau potongan bahasa tertentu merupakan salah satu dari sekian banyak kemungkinan.

Hasil AI merupakan probabilitas. Sering kali, probabilitasnya tinggi, tetapi tetap saja merupakan probabilitas. Hal ini tidak jauh berbeda dengan pikiran manusia, di mana kita sering kali menebak-nebak, atau mengambil sedikit lompatan untuk menyimpulkan sesuatu. Perbedaannya adalah pikiran manusia mampu mengakses banyak informasi lain, seperti situasi, apa yang baru saja terjadi, pemain, dan lokasi untuk menginformasikan tebakan. AI biasanya hanya memiliki satu jenis masukan, dan biasanya hanya satu contoh masukan tersebut. Jadi, sungguh luar biasa bahwa tebakan tersebut sebaik itu, tetapi ingatlah bahwa itu hanyalah tebakan. Seiring dengan kemajuan AI dan kita mulai mempercayainya untuk melakukan operasi yang semakin penting, hal ini penting untuk diingat – AI hanya membuat tebakan terbaiknya.

AI di Masa Depan Anda

Dalam dua bab ini, kami telah mengulas banyak batasan AI, serta berbagai kemungkinan yang dihadirkannya. Hal terpenting yang dapat diambil adalah bahwa AI adalah perangkat lunak yang sangat canggih, bukan kecerdasan yang baru muncul. Sebagai perangkat lunak, cakupan kemampuannya masih terbatas. Dalam cakupan tersebut, AI dapat melakukan beberapa hal yang menakjubkan.

Peran Anda sebagai teknolog konstruksi adalah untuk menentukan cakupan tersebut dan tujuan yang ingin Anda capai dalam cakupan tersebut. Ini biasanya berarti apa yang Anda

inginkan agar AI capai (tujuan), dan tindakan atau fungsi perangkat lunak apa yang akan digunakan untuk mencapainya. Sisanya harus diserahkan kepada mitra Anda, baik secara internal maupun eksternal.

BAB 8
ALAT-ALAT MASA DEPAN

Industri konstruksi telah menghabiskan dekade terakhir untuk mengadopsi teknologi digital secara bertahap. Seiring dengan perubahan yang terjadi di dalam industri, perkembangan eksternal telah meningkatkan jumlah opsi teknologi yang tersedia untuk adopsi di masa mendatang secara drastis. Platform AEC utama, dari Trimble hingga Autodesk telah mengembangkan penawaran terkait konstruksi mereka, dan komunitas startup teknologi konstruksi telah berkembang pesat, menawarkan berbagai macam alat yang jauh lebih luas daripada yang terjadi beberapa waktu lalu. Dalam hubungan klasik ayam dan telur, semakin banyak startup menghasilkan lebih banyak investasi, yang menghasilkan lebih banyak startup.

Di masa lalu, teknologi paling canggih hanya diperuntukkan bagi perusahaan. Bahkan, hampir semua teknologi digital sebelum iPhone dimulai di perusahaan. Bahkan telepon pintar lebih merupakan fenomena perusahaan, karena Blackberry mahal dan terbatas.

Kemudian pada tahun 2007, Apple memperkenalkan iPhone. Tidak seperti produk sebelumnya, iPhone bersifat visual, grafis, dan berfokus pada aplikasi, seperti komputer Macintosh Apple selama bertahun-tahun. Apple dengan cepat memperkenalkan pasar aplikasi, App Store mereka, sebagai cara untuk mendapatkan fungsionalitas yang sama seperti yang Anda harapkan dari aplikasi di komputer desktop atau laptop, yang diperkecil dan dioptimalkan agar dapat bekerja pada layar yang lebih kecil dan daya yang lebih rendah.

Pentingnya perubahan ini tidak dapat dilebih-lebihkan, karena hal ini menciptakan pasar yang sama sekali baru bagi perusahaan rintisan untuk menciptakan aplikasi yang lebih kecil dan dikembangkan dengan cepat. Hal itu, pada gilirannya, melepaskan gelombang kreativitas yang masih kita nikmati. App Store menciptakan saluran distribusi yang memungkinkan siapa pun untuk menciptakan sesuatu dan dengan cepat menyediakannya di hadapan sejumlah pengguna. Moral dari cerita ini adalah Anda dapat mempersiapkan diri untuk teknologi konstruksi masa depan dengan terlibat dengan produk konsumen yang terkadang muncul lebih dulu.

Pada saat yang sama, perkembangan teknis seperti komputasi awan, dan ledakan alat pengembang seperti GitHub, AWS, dan API yang lebih mudah dibuat, semuanya membuat produksi perangkat lunak menjadi jauh lebih murah dan mudah bagi perusahaan yang lebih kecil. Ini berarti bahwa perusahaan rintisan dapat mencoba berbagai hal, seperti perangkat lunak konstruksi, tanpa banyak uang.

Beberapa perusahaan rintisan perangkat lunak ini telah melakukannya dengan sangat baik. Procore pernah menjadi perusahaan rintisan, misalnya, seperti halnya PlanGrid. Akuisisi Plan-Grid oleh Autodesk senilai $875 juta pada tahun 2018 merupakan salah satu momen penting yang mengubah permainan, dengan menunjukkan kepada para kapitalis ventura dan investor malaikat bahwa teknologi konstruksi merupakan tempat yang tepat untuk berinvestasi. Kaustubh Pandya, kepala di Brick & Mortar, salah satu kapitalis ventura

terkemuka yang bergerak di bidang lingkungan binaan, menggambarkan situasi pada tahun 2019 sebagai kombinasi sempurna antara permintaan industri konstruksi, aktivitas perusahaan rintisan, dan minat investor. Jarang sekali kita melihat para pengusaha, pelanggan, dan investor bersatu pada satu titik waktu untuk memajukan industri. Pandya menceritakan bahwa banyak perusahaan konstruksi memandang transformasi digital dan adopsi teknologi sebagai hal yang penting bagi daya saing mereka. Teknologi telah berubah dari sekadar sesuatu yang bagus untuk dimiliki menjadi sesuatu yang wajib dimiliki.

Seorang pakar industri di bidang investasi, Jesse Devitte adalah salah satu pendiri Borealis Ventures dan Building Ventures, yang keduanya mendanai perusahaan perangkat lunak tahap awal di lingkungan binaan. Pada awal tahun 2020, Jesse merasa bahwa perusahaan rintisan AEC masih kekurangan modal, setidaknya sebagian karena proses penjualan perangkat lunak konstruksi dan desain sangat panjang. Setelah berada di posisi meluncurkan perusahaan rintisan perangkat lunak konstruksi, saya cenderung setuju tentang perlunya lebih banyak modal untuk memungkinkan lebih banyak landasan bagi pengembangan penjualan. Sebagian besar industri telah melihat teknologi konsumen memimpin dalam mengubah cara mereka berbisnis, terutama teknologi terkait telepon seperti aplikasi, pesan, dan foto. Konstruksi jelas harus melangkah lebih jauh, karena begitu banyak pekerjaan sehari-hari berbasis kertas, tetapi tren umumnya sama di seluruh perekonomian.

Tingkat teknologi konsumen yang tinggi ini berarti bahwa banyak hal yang akan berdampak pada lokasi kerja di masa depan ada di tangan Anda saat ini, dan relatif mudah untuk dicoba. Itu adalah perubahan penting dari masa lalu yang memiliki dua implikasi besar. Yang pertama adalah bahwa setiap orang bisa mendapatkan pengalaman langsung dengan banyak produk dan teknologi baru tanpa mengeluarkan uang; dan implikasi kedua adalah bahwa tidak diperlukan gelar khusus, atau banyak pelatihan untuk menggunakan sebagian besar produk. Cara umum kerja satu antarmuka produk cloud akan mengajarkan Anda sebagian besar hal yang perlu Anda ketahui tentang produk cloud lainnya. Kita semua menggunakan kosakata antarmuka umum yang sama dari tombol, bidang, gesekan, dan klik.

Namun, ada beberapa teknologi yang benar-benar baru yang memerlukan sedikit penjelasan lebih lanjut. Di bagian bab ini, kita akan mengulas enam teknologi yang akan semakin penting di masa depan – tiga di antaranya merupakan perpaduan antara perusahaan dan konsumen: realitas virtual, realitas tertambah, dan pemindaian 3D berkualitas tinggi. Kemudian kita akan melihat tiga teknologi yang tidak benar-benar berorientasi pada konsumen, tetapi akan sangat penting: sensor, IoT, dan kembaran digital generasi berikutnya.

8.1 REALITAS VIRTUAL (VR)

Seperti banyak teknologi yang kini mulai populer, realitas virtual bukanlah hal baru. Pada tahun 1960-an, cermin dan televisi CRT besar digunakan untuk mencoba menciptakan rasa "berada di sana" yang mendalam. Sama seperti AI, ada beberapa upaya sebelumnya untuk menciptakan realitas virtual yang tidak berhasil karena teknologi yang mendasarinya tidak ada.

Pertama, mari kita definisikan apa yang kita maksud dengan realitas virtual. Realitas virtual menggunakan beberapa layar atau proyektor untuk menciptakan ilusi bahwa pengguna berada di dunia virtual. VR menghalangi dunia nyata dan menggantinya dengan dunia virtual. VR dapat dicapai dengan video 360 derajat, atau dengan grafik komputer – intinya adalah ilusi bahwa Anda berada di suatu tempat yang sebenarnya tidak Anda alami. Hal ini biasanya dicapai dengan headset, di mana setiap mata memiliki layarnya sendiri, tetapi ada contoh yang menggunakan proyektor dan kacamata khusus di ruangan yang dibangun khusus, yang disebut CAVE.

Beberapa pembaca mungkin ingat bahwa pada tahun 1990-an ada headset konsumen dan instalasi realitas virtual berbasis lokasi di arena permainan dan beberapa tujuan liburan. Hal ini memiliki sejumlah masalah, salah satunya adalah grafiknya yang buruk dan membuat orang sakit. Selama bertahun-tahun, industri telah menyadari bahwa apa yang membuat orang sakit berkaitan dengan cara kerja mata manusia. Bagian tengah mata Anda disebut "fovea", dan di sanalah sebagian besar penginderaan warna dan hampir semua hal yang sangat mendetail terjadi. Fovea adalah pusat bidang penglihatan Anda, dan dapat mendeteksi perubahan yang terjadi sekitar 30 kali per detik. Kedengarannya cepat, dan memang begitu. Namun, fovea hanya mencakup bagian tengah perhatian Anda.

Ternyata bagian retina lainnya bekerja dengan cara lain. Sementara fovea memiliki "kerucut" berwarna yang melihat dengan sangat detail, penglihatan tepi terdiri dari batang yang berwarna hitam dan putih, yang berfungsi untuk mendeteksi gerakan. Jika Anda memikirkan cara kerja penglihatan dan perhatian manusia dalam kehidupan nyata, itu masuk akal. Anda memiliki sensor gerakan di bagian tepi dan detektor warna yang sangat detail di pusat perhatian Anda – jadi penglihatan tepi Anda menangkap gerakan, lalu Anda mengalihkan mata untuk melihat apakah penyebab gerakan itu berbahaya atau tidak.

Pernahkah Anda memperhatikan bahwa Anda dapat melihat televisi berkedip-kedip dari sudut mata Anda, tetapi tidak saat Anda melihatnya langsung? Penginderaan gerakan yang ditingkatkan oleh penglihatan tepi adalah alasannya, dan itu berarti VR harus setidaknya 60 bingkai/detik untuk menghindari penyakit. Semua VR modern setidaknya memiliki level ini, jadi kemungkinan Anda untuk sakit jauh lebih kecil daripada sebelumnya. Alasan lain mengapa orang jatuh sakit adalah karena kita tidak suka jika satu kelompok indera memberi tahu kita bahwa kita sedang bergerak dan kelompok lain memberi tahu kita bahwa kita tidak sedang bergerak. Inilah sebabnya mengapa banyak orang mabuk perjalanan saat membaca, dan itulah sebabnya VR yang membuat Anda bergerak melalui pemandangan virtual tanpa benar-benar bergerak membuat sebagian orang merasa mual. Jadi, kami tidak melakukan itu, kami memastikan saat Anda perlu bergerak, Anda melakukannya dengan cepat, atau setidaknya dengan cara yang berada di bawah kendali Anda.

Hal ini dan banyak trik lainnya telah menjadikan realitas virtual pada tahun 2020 sebagai pengalaman yang jauh lebih baik daripada apa pun di masa lalu. Pada saat yang sama, harga perangkat keras, dan kualitas pengalaman terus meningkat secara dramatis. Secara keseluruhan, hambatan untuk mengadopsi VR telah menurun drastis, yang menyisakan

pertanyaan besar tentang apa sebenarnya manfaat VR, dan di mana hal itu akan mengubah konstruksi dalam waktu dekat?

Mengapa VR

Realitas virtual adalah pengalaman yang unik, karena ia menghalangi dunia, dan menggantinya dengan sesuatu yang lain. Artinya, Anda dapat pergi ke mana saja, menciptakan lingkungan apa saja, dan benar-benar membuat orang percaya bahwa mereka berada di dunia baru ini. Itu luar biasa hebatnya, dan baru sekarang kita mulai melihat VR memenuhi potensinya. Ada empat hal yang harus Anda pahami tentang efektivitas realitas virtual: kehadiran, orang pertama, fisika fleksibel, dan analitik.

Kehadiran Versus Imersi

Kita sering menyebut realitas virtual imersif, karena pengguna tenggelam dalam konten. Ini hebat, terutama karena VR benar-benar imersif. Desainer mampu memblokir bukti apa pun bahwa dunia virtual itu tidak nyata, sehingga pengguna jauh lebih tenggelam daripada, katakanlah, bioskop.

Imersi adalah tentang teknologi, tentang peralatan. Yang benar-benar penting adalah bahwa imersi lengkap ini dapat mengarah pada apa yang kita sebut "kehadiran," di mana pengguna benar-benar percaya pada ilusi bahwa mereka berada di dunia virtual. Ini adalah fenomena khusus yang tampaknya berlawanan dengan intuisi pada awalnya, karena pengguna baru saja masuk ke ruangan normal dan mengenakan headset. Namun, pikiran manusia sangat pandai bertindak seolah-olah apa yang dilihatnya nyata, terutama ketika ada cerita yang harus diikuti, atau ada sedikit tanda bahaya.

Di New York ada perusahaan VR/AR besar yang disebut Glimpse Group. Pada saat penulisan ini, saya menjalankan unit yang menciptakan pengalaman VR untuk pelatihan – kami membuat VR dan AR untuk manufaktur, keselamatan, konstruksi, universitas, dan banyak lagi. Untuk meyakinkan klien potensial tentang kekuatan VR, kami sering menjalankan pengalaman "Glimpse Tower". Dalam aplikasi ini, pengguna dimasukkan ke dalam lift yang akan membawa Anda ke lantai 30, di mana Anda kemudian diminta untuk berjalan di atas papan pendek di atas ketinggian yang luar biasa itu untuk mengganti bola lampu. Saya secara pribadi telah menunjukkan pengalaman itu kepada lebih dari 100 orang, dan setiap orang bereaksi dengan tingkat kecemasan tertentu, beberapa bahkan menolak untuk berjalan melintasi papan itu. Orang-orang percaya pada ilusi berada di lantai 30, bahkan ketika mereka berada di kantor mereka sendiri.

Perangkat keras menghalangi dunia luar, dan konten menciptakan ilusi bahwa pengguna berada di suatu tempat yang tidak mereka ketahui, menciptakan "kehadiran", yang bersifat psikologis. Kekuatan kehadiran dalam realitas virtual tidak terikat pada grafik berkualitas tinggi seperti yang mungkin dipikirkan orang. Faktanya, berbagai penelitian telah menunjukkan bahwa pikiran cukup pandai mengatasi grafik yang dihasilkan komputer jika ada alasan kuat untuk percaya bahwa adegan itu nyata, atau setidaknya bisa jadi nyata. Terutama ketika ketakutan atau kecemasan terlibat, tetapi juga ketika hubungan manusia terlibat.

Seperti yang akan dipahami oleh siapa pun yang telah menonton film animasi modern, kita dapat mengembangkan rasa sayang terhadap karakter yang digambar komputer, bahkan

ketika mereka tampak seperti balon anak-anak dengan mata. Apa yang dipelajari studio-studio ini sejak awal adalah bahwa jika gerakannya tampak manusiawi, dan suaranya nyata, kita dapat mengatasi sisa presentasi karakter yang sedikit kurang tampak manusiawi. Ini bahkan lebih berlaku untuk model mesin dalam VR, di mana kita baik-baik saja dengan tidak adanya fotorealistis.

Presence memungkinkan Anda untuk beroperasi dalam VR seolah-olah itu adalah kehidupan nyata, dengan keterlibatan yang sama, dan keyakinan bahwa tindakan dan kesalahan akan memiliki konsekuensi nyata yang akan terjadi jika melakukan tindakan atau tugas dalam kehidupan nyata. Kita dapat menempatkan orang-orang di atas gedung yang setengah jadi dan menunjukkan kepada mereka seperti apa bekerja di sana. Kami membuat tiruan bangunan sebelum pekerjaan apa pun dimulai, dan memperbolehkan pemilik merasakan desainnya secara langsung.

Orang Pertama

Kehadiran sangatlah kuat karena dalam realitas virtual, tidak seperti video atau pemirsa model BIM, Anda adalah aksinya, itu terjadi pada Anda. Perasaan berada sebagai orang pertama sering kali aneh dalam video, karena Anda tidak dapat mengendalikan ke mana kamera bergerak – jadi ketika Anda melihat video melalui mata pengguna, mungkin sulit untuk memprosesnya. Video permainan tembak-menembak orang pertama membuktikan betapa anehnya hal itu.

Dalam realitas virtual, ada dua hal berbeda yang membuat orang pertama jauh lebih alami dan kuat. Yang pertama adalah bahwa realitas virtual mengasumsikan Anda adalah orang yang mengendalikan "kamera" – gerakan kepala Anda memutuskan apa yang akan Anda lihat selanjutnya, sama seperti dalam kehidupan nyata. Kemampuan untuk memutuskan apa yang akan Anda lihat selanjutnya merupakan kontributor besar bagi kehadiran, dan itu adalah bagian dari apa yang membuat VR benar-benar berguna sebagai alat untuk melakukan pekerjaan, dan untuk pelatihan. Hal kedua yang berbeda adalah bahwa VR menurut definisinya adalah pengalaman 360 derajat penuh – Anda dapat berbalik, melihat ke atas, ke bawah, ke mana pun.

Anda tidak berada di luar suatu adegan sambil melihat ke dalam, Anda berada di dalam adegan tersebut. Dan itu berarti Anda adalah pusat aksi, dan itu terjadi pada Anda. Kami tahu dari pengalaman bertahun-tahun dengan pelatihan dan pengalaman VR lainnya bahwa kami memproses berbagai hal dengan sangat berbeda ketika hal itu berpotensi memengaruhi kami. Keterlibatan tidak berarti keterlibatan, karena banyak acara televisi dan film melibatkan kita sepenuhnya tanpa benar-benar melibatkan kita. Pada saat menonton, kita tidak benar-benar berpikir bahwa apa yang terjadi dalam acara dan film yang menarik ini akan berdampak pada kehidupan kita, pada tubuh kita sendiri. Ini karena kita menyaksikan aksi yang terjadi melalui jendela.

Sebaliknya, realitas virtual menempatkan pengguna tepat di dalam aksi, dan meskipun pengguna tidak pernah dalam bahaya apa pun, pikiran terprogram untuk tidak mengambil risiko, jadi kita merespons seolah-olah bahaya itu sampai batas tertentu nyata. Namun, ini

bukan hanya tentang rasa takut dan bahaya. Saat kita melakukan sesuatu dalam VR, kita mempelajarinya lebih baik daripada jika kita hanya membaca atau menontonnya.

Pengalaman orang pertama dalam lingkungan 3D yang mendalam ini memiliki dampak lain yang sangat penting. Kita dapat melihat lebih banyak pemandangan, memahaminya dengan lebih baik, saat kita berada di dalamnya. Lokasi kerja akan jauh lebih masuk akal saat Anda berada di sana, daripada jika Anda hanya melihat gambar. Kokpit derek yang berada 30 kaki di udara terasa sangat berbeda saat Anda berada dalam versi virtual kokpit tersebut daripada jika Anda menonton video. Model BIM yang dapat Anda lihat akan berkomunikasi jauh lebih menyeluruh daripada model yang dapat Anda lihat dalam 2D.

Fisika Fleksibel

Sedangkan kehadiran dan orang pertama menjadikan realitas virtual sebagai cara yang ampuh untuk mengoperasikan dan melatih proses fisik, fakta bahwa realitas virtual sepenuhnya dihasilkan oleh komputer berarti tidak ada yang tidak Anda pilih untuk disertakan. Anda dapat membuat gravitasi hanya berfungsi untuk hal-hal tertentu, misalnya. Atau Anda dapat membuat benda saling menempel, mengapung, apa pun.

Kita baru mulai memahami bagaimana ini dapat bermanfaat, karena kita sedikit terpaku pada hal-hal yang berfungsi seperti yang selalu terjadi di dunia nyata. Ini adalah pola yang berulang untuk setiap media baru: kita menggunakan media baru seperti kita menggunakan media lama, hingga kita benar-benar memahami hal baru dan mulai berinovasi.

Bayangkan jika rapat perencanaan penarikan Anda memungkinkan tim untuk menarik video 360^{o} penuh dari lokasi kerja untuk menunjukkan suatu hal, membawa seluruh tim ke tempat kejadian yang direkam untuk melihat bagaimana keselamatan diterapkan, misalnya. Atau jika Anda dapat menggunakan suara untuk membuat catatan dalam rapat tersebut, dan setiap catatan secara otomatis dimasukkan ke dalam basis data yang kemudian dapat disortir, diedit, dan dikumpulkan untuk digunakan dalam laporan dan hal-hal lainnya. Catatan-catatan tersebut juga dapat dikumpulkan ke dalam beberapa dinding untuk penjadwalan dan persentase diskusi yang selesai, di mana setiap gerakan catatan tempel segera mengubah jadwal, dan seterusnya.

Pembengkokan realitas agar sesuai dengan proses Anda berpotensi mengubah proses tersebut secara drastis. Tiba-tiba informasi segera tersedia, di mana-mana, dan dalam bentuk yang jauh lebih lengkap dan mendalam yang sangat membantu. Anda dapat memiliki sejumlah asisten suara untuk membantu setiap peserta dan menjalankan rapat, dengan semua yang direkam dan diedit sehingga komitmen dicatat, dan diterjemahkan ke dalam pesanan material atau format terkait proses lainnya.

Analitik

Semua hal dalam realitas virtual hadir melalui komputer, yang berarti Anda dapat melacak semuanya. Ini tidak berarti semuanya dilacak secara otomatis, hanya saja Anda dapat melacaknya jika ada alasan untuk itu. Misalnya, Anda dapat melacak ke mana pengguna melihat, di mana mereka berada dalam suatu adegan, berapa lama waktu yang dibutuhkan untuk melakukan sesuatu, seberapa baik hal-hal tersebut diselesaikan, dan sebagainya.

Realitas virtual memungkinkan pengguna dan manajer untuk memahami secara tepat kinerja pengguna, yang dapat jauh lebih baik daripada pengamatan yang lebih longgar dan kurang tepat yang sering kali merupakan hal terbaik yang dapat Anda lakukan di luar VR.

8.2 REALITAS TERTAMBAH (AR)

Jika VR menempatkan pengguna dalam lingkungan virtual, AR memungkinkan pengguna untuk membawa objek virtual ke dunia nyata. Jika VR memerlukan headset penuh dan menghalangi seluruh dunia, AR menggunakan sepasang kacamata, tablet, atau ponsel untuk melihat objek virtual ini, dan menurut definisi terintegrasi dengan seluruh dunia.

Definisi AR yang lebih formal adalah penggunaan lensa atau perangkat seluler untuk melapisi dunia nyata dengan informasi dan konten. Konten ini berorientasi pada objek nyata di luar angkasa, baik melalui pemetaan spasial, atau melalui penanda khusus yang dikenali oleh sistem AR. Realitas tertambah juga disebut sebagai "realitas campuran", tetapi perbedaan antara keduanya adalah istilah pemasaran, bukan teknologi yang sebenarnya.

Realitas tertambah berfungsi karena kamera, atau sensor lain seperti LiDAR, menemukan sesuatu di lingkungan yang memungkinkannya untuk "berlabuh" dan mendapatkan beberapa tingkat arah di dalam ruangan.

Mungkin AR tampak lebih mudah daripada VR, jadi seharusnya lebih maju dan lebih bermanfaat, tetapi sebenarnya sangat sulit untuk menempatkan elemen virtual ke dalam ruang yang tidak dibuat oleh pengembangnya sendiri.

Ingat kembali pada bab kecerdasan buatan yang kita bahas tentang kasus-kasus ekstrem. Dalam kasus AR, sebagian dari kompleksitas yang sama sedang terjadi; karena AR seharusnya berfungsi di mana saja, itu berarti Anda akan memiliki berbagai macam permukaan, pencahayaan, dan objek yang memungkinkan di dalam ruangan yang harus dipahami dan diintegrasikan oleh komputer sehingga dapat meletakkan berbagai hal di depan objek-objek ini.

Realitas tertambah telah terus menjadi lebih canggih karena teknologi telah berkembang melalui empat fase berbeda hingga saat ini, dengan setidaknya dua fase yang diharapkan di masa mendatang.

Fase 1: AR berbasis penanda

Kamera telepon pintar memindai suatu pemandangan, dan telepon mengenali sesuatu, seperti kode QR, atau gambar yang telah dilatih untuk dikenali oleh sistem. Biasanya yang akan terjadi kemudian adalah satu gambar, video, atau gambar 3D akan muncul di depan, atau di samping, penanda. Sejak tahun 2013, hal semacam ini digunakan oleh arsitek yang paham teknologi untuk memamerkan desain dalam presentasi.

AR berbasis penanda masih digunakan karena murah dan mudah dijalankan. Misalnya, Glimpse memiliki produk yang disebut "Post-Reality," yang memungkinkan pengguna untuk menciptakan pengalaman AR dengan cepat dan mudah tanpa perlu membuat kode.

Fase 2: Visi komputer/SLAM

Pendekatan yang lebih canggih mengambil gambar dari kamera tablet atau ponsel, dan menerapkan semacam algoritme yang dikenal sebagai "lokasi dan pemetaan simultan"

(SLAM), yang berarti ia mengidentifikasi permukaan datar, tepi, dan fitur lain dalam gambar yang seharusnya datar. Melalui fitur gambar ini, sistem menentukan tempat untuk mengintegrasikan objek virtual guna memberikan pengalaman AR.

Aplikasi berbasis SLAM telah digunakan secara luas dalam ritel, dengan merek seperti Ikea menerbitkan aplikasi telepon pintar yang memungkinkan pengguna untuk memilih furnitur dan menempatkan gambarnya di lantai di rumah atau kantor mereka yang sebenarnya.

Fase 3: Headset

Pada tahun 2016, Microsoft merilis Hololens, headset realitas tertambah yang merupakan keajaiban teknologi sejati. Hololens menggunakan penyempurnaan dan perluasan teknologi Kinect mereka, yang awalnya dikembangkan untuk Xbox. Hololens memetakan secara digital ke mana pun Anda melihat dalam pemandangan, yang memungkinkan Anda mengimpor dan menempatkan objek virtual ke dalam pemandangan tersebut. Ia menyimpan hasil pemindaian ruangan, dan objek yang telah Anda tempatkan di dalamnya, jadi jika Anda pergi dan kembali, selama Anda mengenakan Hololens, Anda dapat melihat objek virtual yang telah Anda tempatkan di sana.

Pada tahun 2018, sebuah perusahaan rintisan dari Florida, Magic Leap, memperkenalkan headset AR mereka sendiri, Magic Leap One. Secara efektif, headset ini sama dengan Hololens dalam hal kinerja dan harga, sekitar Rp. 30.000.000. Seperti Hololens, bidang pandang yang sempit secara signifikan membatasi kegunaan Magic Leap One. Bidang pandangnya sekitar seperlima hingga seperempat dari bidang pandang alami pengguna, menyebabkan banyak pengguna baru memutuskan untuk tidak menggunakan teknologi tersebut.

Sebagian besar, headset AR yang telah dirilis pada pertengahan tahun 2020 terlalu mahal dan terbatas untuk diadopsi secara luas. Pengecualian yang mungkin adalah Trimble Connect, yang menggabungkan helm dengan Hololens yang telah dimuat sebelumnya dengan perangkat lunak Trimble seperti SiteVision.

Fakta bahwa sumber daya Microsoft, dan Magic Leap yang didanai sangat besar, hanya mampu menciptakan perangkat dengan bidang pandang terbatas ini menggambarkan betapa sulitnya menciptakan sistem yang dapat secara efektif mengintegrasikan konten virtual dengan dunia nyata. Namun, secara luas diharapkan bahwa headset dari Apple, Facebook, dan lainnya akan jauh lebih ramah pengguna dalam 2–5 tahun ke depan.

Fase 4 : LiDAR

Fungsi utama semua pendekatan AR adalah memahami apa yang ada di lingkungan, sehingga dapat dipetakan ke elemen virtual apa pun yang akan diperkenalkan. Sementara SLAM dan pendekatan visi mesin lainnya dapat efektif, semuanya menggunakan beberapa bentuk bagi AI untuk menginterpretasikan gambar datar. Ini berarti mereka perlu meluangkan waktu untuk memahami di mana tepi dan permukaan berada, yang dapat berdampak negatif pada pengalaman pengguna. Kinect/Hololens sedikit lebih baik dengan memanfaatkan sejumlah teknologi, termasuk memproyeksikan kisi titik inframerah yang digunakannya untuk memetakan ruang.

Namun, tidak satu pun dari pendekatan ini akan memiliki akurasi dan kecepatan LiDAR, yang merupakan akronim untuk Light Distancing and Ranging, yang pada dasarnya menggunakan laser untuk melakukan radar. LiDAR telah menjadi bagian besar dari bagaimana perusahaan kendaraan otonom seperti Waymo milik Google telah melatih mobil mereka untuk mengemudi, karena memberikan gambaran dunia yang tepat.

Dan tentu saja, perusahaan konstruksi telah menggunakan pemindaian laser selama bertahun-tahun, untuk survei dan penangkapan realitas. Perbedaan besarnya sekarang adalah perangkat konsumen memiliki kemampuan untuk memindai dunia dengan laser dan mengetahui secara pasti dimensi ruangan dan segala isinya, serta di mana perangkat tersebut berada dalam model ruangan tersebut.

Shapr3D telah mulai menggunakan sensor LiDAR baru Apple di iPad pro mereka untuk memindai ruangan dan menggabungkannya langsung ke dalam proses desain, dengan presisi dan kecepatan yang tidak mungkin dilakukan beberapa waktu lalu.

Potensi yang dimilikinya untuk augmented reality yang kuat dan benar-benar bermanfaat sangat besar. Kita akan melihat dalam beberapa tahun mendatang produk apa yang sebenarnya akan hadir di pasaran. Mudah untuk membayangkan AR digunakan untuk inventarisasi otomatis, untuk tampilan model BIM yang sangat akurat terhadap status pekerjaan terkini yang sedang dilakukan, di antara banyak kemungkinan lainnya.

Keempat fase ini secara kasar memetakan apa yang telah kita lihat dalam AR dalam beberapa tahun terakhir, dan masing-masing fase ini masih ada di pasaran – tidak mengherankan fase-fase awal cenderung lebih murah dan lebih mudah diakses, sedangkan fase-fase selanjutnya, lebih mahal tetapi lebih mampu. Dalam beberapa tahun mendatang, ada dua perkembangan yang sangat signifikan yang diharapkan: kacamata AR dan cloud AR. Mana yang akan muncul "lebih dulu" bukanlah hal yang penting, karena keduanya akan diluncurkan dalam bentuk yang kurang canggih dan akan berkembang dengan cepat.

Tahap 5: AR Cloud

"Cloud" sebenarnya hanyalah sekumpulan server yang memungkinkan informasi dan layanan perangkat lunak diakses dari mana saja. Pada tahun 2017, Ori Inbar, pendiri AWE dan Super Ventures, mengemukakan gagasan untuk menggunakan cloud guna menyediakan layanan augmented reality di mana saja. Namun, gagasannya melampaui layanan cloud standar, yakni menyertakan peta dunia yang sangat terperinci, yang dapat diakses kapan saja, di mana saja.

Tujuan dari cloud AR adalah menciptakan dunia kembaran digital, yang dapat dilihat melalui ponsel, kacamata, atau alat penampil lainnya. Dengan cloud AR, pengguna dapat datang ke tempat yang belum pernah mereka kunjungi sebelumnya, dan dengan mengakses cloud AR, mereka dapat langsung berintegrasi dengan ekonomi dan masyarakat setempat – semuanya dari ponsel atau kacamata AR mereka.

Teknologi ini belum sepenuhnya berhasil, tetapi semua orang mulai dari Google hingga Magic Leap dan setidaknya satu konsorsium teknisi internasional bekerja keras untuk mewujudkannya. Manfaat potensial bagi konstruksi adalah aplikasi AR yang jauh lebih canggih

di lokasi kerja, dengan kemungkinan bahwa pembuatan aplikasi AR baru akan menjadi jauh lebih mudah di masa mendatang.

Fase 6: Kacamata AR

Semua teknologi AR sejauh ini mengasumsikan pengalaman menonton yang mengharuskan pengguna untuk menghentikan sesuatu yang lain dan melihat melalui headset khusus, atau ponsel atau tablet mereka. Ini hebat, dan akan menjadi lebih hebat lagi di tahun-tahun mendatang.

Namun, visi utama untuk AR adalah menjadi sesuatu yang berfungsi sepanjang waktu. Solusi AR sehari-hari yang diharapkan industri adalah sepasang kacamata yang Anda kenakan seperti kacamata hitam, dan dapat dikenakan sepanjang waktu. Beberapa perusahaan telah mengindikasikan, atau dikabarkan sedang mengerjakan ini, tetapi ini adalah tantangan yang sangat sulit. Tiga alasannya:

1. ***Daya pemrosesan:*** Memahami suatu pemandangan dan kemudian menempatkan sesuatu dalam pemandangan itu membutuhkan banyak sekali perhitungan angka. Menempatkan semua itu di hidung seseorang tidaklah mudah – rata-rata berat kacamata sekitar satu ons, sedangkan Hololens 2 beratnya 1,28 pon, atau lebih dari dua puluh kali berat kacamata.
2. ***Panas***: Berapa pun daya pemrosesan yang akhirnya digunakan, akan menghasilkan lebih banyak panas daripada yang biasa ditanggung pengguna. Itu berarti beberapa desain cerdas akan menjauhkan panas dari wajah.
3. ***Sinar matahari***: Headset AR saat ini tidak berfungsi dengan baik di bawah sinar matahari. Cahaya terang akan memudarkan lensanya. Jelas ini perlu dipecahkan, tetapi lebih sulit daripada kedengarannya: terlalu terang akan menyakiti mata pengguna, terlalu redup akan memudar. Salah satu solusinya adalah membuat piksel "hitam" yang menghalangi cahaya, yang lain adalah membuat warna datar yang menciptakan gambar dengan memantulkan cahaya, tetapi ini sekali lagi sangat sulit. Kita harus melihat bagaimana ini dipecahkan saat produk ini dipasarkan, kemungkinan pada tahun 2021 dan seterusnya.

Realitas tertambah akan menjadi salah satu teknologi yang paling penting dan paling meluas di lokasi kerja dalam beberapa tahun mendatang. Namun, agar ini terjadi, teknologi ini harus terlebih dahulu diterima secara luas sebagai produk konsumen. Ini kemungkinan berarti bahwa aplikasi AR akan mudah digunakan dan dirancang untuk kurva pembelajaran yang pendek.

8.3 PEMINDAIAN 3D

Pemindaian laser telah menjadi bagian dari konstruksi selama bertahun-tahun. Perusahaan seperti Leica, Matterport, Faro, dan lainnya memiliki teknologi luar biasa yang memungkinkan pemindaian yang sangat akurat di hampir semua lokasi. Perangkat lunak untuk memanfaatkan pemindaian berbasis laser ini juga terus mengalami peningkatan.

Apa yang kami lihat di pasaran adalah perluasan dari teknologi visi komputer yang sama yang telah kami bahas di atas di bagian AR – pemindaian berkualitas tinggi yang semakin

banyak berasal dari perangkat kelas konsumen. Sebagai contoh, Matterport, pembuat berbagai solusi pemindaian dan fotografi untuk real estat dan lingkungan binaan, merilis aplikasi seluler pada bulan April 2020. Aplikasi ini memungkinkan konsumen untuk memindai ruangan dengan petunjuk yang sangat mudah digunakan, dan membuat model ruangan tersebut.

Ini dikenal sebagai fotogrametri – menggunakan foto untuk membuat model 3D suatu ruang atau objek. Ini adalah jenis hal yang biasanya dilakukan oleh pemindaian laser, kecuali dalam fotogrametri Anda memiliki warna dan pola, karena tentu saja itu adalah foto. Bahkan fotogrametri terbaik pun tidak akan menghasilkan model yang seakurat pemindaian laser, tetapi untuk banyak hal itu tidak penting.

Perusahaan selain Matterport telah merilis aplikasi pemindaian dan fotogrametri serupa, meskipun biasanya tidak dikemas dengan baik. Hal penting yang perlu dipahami adalah bahwa visi mesin semakin baik dalam "melihat" dunia, dan akan terus menjadi lebih baik.

Ingat bahwa aturan umum tentang AI, dan visi mesin adalah contoh hebat dari AI praktis, adalah bahwa berlalunya waktu pasti akan menghasilkan kinerja yang lebih baik, karena masing-masing dari tiga unsur AI akan terus meningkat: kita akan terus mendapatkan prosesor yang lebih cepat, lebih banyak data, dan algoritme yang lebih baik.

Dan seiring dengan semakin baiknya visi mesin, ia akan mulai mencapai titik kritis di mana ia tidak hanya lebih baik, tetapi juga berbeda. Perangkat genggam Anda, dan aplikasi yang dapat didukungnya, tidak akan hanya dapat melihat permukaan datar untuk AR. Aplikasi akan melampaui pembuatan peta berwarna 3D dari sebuah ruangan, dan akan mulai mengidentifikasi apa yang sedang dipindai. Mereka akan melampaui bentuk untuk mengidentifikasi produk, rakitan, dan lainnya yang spesifik. Google Lens sudah melakukan ini dengan barang-barang konsumen, dan dengan cepat menjadi lebih baik.

Pemindaian semacam ini dapat menciptakan kemampuan inventaris yang telah kita bahas sebelumnya, tetapi juga dapat mengenali bahaya di lokasi kerja dan mendeteksi masalah dalam pekerjaan yang dilakukan, termasuk varians dari desain yang disetujui. Faktanya, ini sudah dilakukan pada tingkat yang jauh lebih sederhana oleh perusahaan yang menggunakan visi mesin untuk menegakkan jarak sosial di lokasi kerja setelah Covid-19.

Bab-bab AI menguraikan beberapa batasan AI, dan pemindaian semacam ini dan pemahaman dunia nyata akan menunjukkan batasan tersebut karena hal-hal umum mudah diidentifikasi tetapi kasus-kasus ekstrem terus membuat sistem frustrasi. Dalam waktu dekat, fotogrametri dan pemindaian berbantuan LiDAR akan dapat mengidentifikasi beberapa hal, dan akan salah mengenali yang lain. Namun, jalur teknologi ini saat ini menjamin seseorang akan menciptakan aplikasi yang luar biasa. Mungkin bahkan Anda.

Teknologi Berbasis Perusahaan

Kode batang dikembangkan pada tahun 1950-an untuk melacak barang-barang industri, dan akhirnya mencapai kesuksesan komersial pada tahun 1970-an dengan pemindai kode batang supermarket yang sudah dikenal. Kebutuhan untuk memahami lokasi produk di gudang dan rantai pasokan telah menghasilkan berbagai macam teknologi, mulai dari RFID

hingga iBeacon dan lainnya. Beberapa di antaranya melibatkan pemindaian sesuatu pada barang, yang lain bekerja dengan memancarkan sinyal dari barang tersebut.

Pada saat yang sama, manusia telah merasakan lingkungan mereka setidaknya selama satu abad dengan termostat, dan menggunakan informasi yang dihasilkan sensor ini untuk mengendalikan karakteristik lingkungan tersebut. Kita memiliki sensor cahaya, sensor panas, sensor kelembapan, sensor gerak, sensor kimia, dan sebagainya.

Kedua jenis teknologi yang berbeda ini sama-sama tentang pemahaman lingkungan: dalam kasus kode batang dan RFID, ini tentang pemahaman pergerakan benda melalui lingkungan; dalam kasus sensor, ini tentang pemahaman keadaan lingkungan. Di masa lalu, masing-masing sensor ini merupakan silo data, biasanya hanya satu mesin yang menangkap beberapa jenis data yang kemudian digunakan untuk satu jenis penggunaan, seperti menghitung barang yang terjual, atau barang di pusat distribusi, atau menjaga suhu, kelembapan, atau cahaya pada tingkat yang sesuai.

Jaringan yang kita bahas di Bab 3 telah mulai merangkai sensor-sensor ini, hingga seseorang membuat pengamatan yang cemerlang bahwa sebenarnya tidak ada perbedaan antara internet yang digunakan orang untuk saling mengirim email dan informasi lainnya, dan internet yang digunakan mesin untuk saling mengirim data. Jadi istilah "Internet of Things" dicetuskan pada tahun 1999, oleh seorang eksekutif di Proctor & Gamble. Bahkan, tampaknya perangkat pertama yang terhubung ke internet adalah mesin Coca-Cola di Universitas Carnegie-Mellon.

Internet of Things, atau IoT, lebih dari sekadar koneksi data yang menghasilkan sensor. Inti dari IoT adalah pengetahuan waktu nyata tentang apa yang terjadi di seluruh lokasi kerja, pabrik manufaktur, atau rantai pasokan. Aliran sensor terperinci dan waktu nyata itu menghasilkan banyak sekali data, yang selanjutnya dimasukkan ke dalam model pembelajaran mesin. Pada gilirannya, model-model ini dapat memprediksi kapan mesin-mesin kemungkinan perlu diperbaiki, sebuah layanan yang telah disediakan General Electric, Siemens, dan perusahaan-perusahaan industri lainnya untuk mesin pesawat terbang dan pembangkit listrik selama bertahun-tahun. Semakin akurat data yang dimasukkan ke dalam model-model prediktif ini, semakin akurat dan spesifik prediksi mereka, sehingga ada gerakan yang sedang berlangsung untuk menempatkan lebih banyak sensor pada lebih banyak hal. Manfaatnya bagi rantai pasokan dan proses produksi sangat signifikan. Kita mulai melihat beberapa pendekatan yang sama dalam konstruksi, jadi ada baiknya untuk melihat sekilas teknologi-teknologi ini sehingga Anda memahaminya saat muncul:

1. RFID. Ini adalah singkatan dari “Radio Frequency Identifier” – ini sering digunakan oleh konsumen untuk jalan tol, misalnya di timur laut AS sebagai “EZ-Pass.” Tag RFID menggunakan chip yang menyerap energi radio dari pemancar yang relatif dekat, kemudian memancarkan kembali ID unik tag tersebut. RFID berguna karena tag itu sendiri tidak perlu menyimpan energi apa pun, dan dapat dibaca dari jarak tertentu. Namun, tag RFID dapat berharga lebih dari Rp. 100/buah, yang menambah biaya yang signifikan selama masa pakai suatu proses, dan banyak RFID yang cocok untuk aplikasi konstruksi akan jauh lebih mahal.

2. iBeacons. Memahami di mana suatu barang berada di dalam suatu ruang bisa jadi sulit. GPS tidak terlalu akurat, dan sering kali tidak dapat menembus ke dalam bangunan. iBeacons dan Eddystone adalah dua versi teknologi yang menggunakan sinyal Bluetooth untuk memungkinkan lokasi barang yang jauh lebih akurat di dalam suatu ruang.
3. NFC. Near-Field Communication memungkinkan telepon pintar untuk membayar di kasir tanpa kontak. NFC sering digunakan untuk membuat koneksi pertama antara perangkat, yang kemudian digantikan oleh komunikasi yang lebih kuat, seperti Bluetooth.
4. Akselerometer. Memahami apakah barang yang rusak telah salah penanganan selama pengiriman dapat menjadi hal yang berharga. Akselerometer adalah sensor yang mencatat gerakan tiba-tiba, seperti terjatuh. Telepon pintar memiliki akselerometer yang memungkinkan perangkat merasakan gerakan, tetapi sebagian besar akselerometer IoT akan disetel untuk mencari kejadian yang berpotensi merusak.

Semua sensor yang disebutkan berhubungan dengan benda yang bergerak; ada banyak sensor lain untuk ruangan dan lokasi tetap, seperti sensor gas, sensor jarak, sensor gerak, penghitung orang, dan banyak lagi.

Menyatukan Gambaran Baru

Tidak mudah untuk menyatukan sumber data ini menjadi sesuatu yang berguna, dan tidak selalu lebih banyak informasi lebih baik. Misalnya, mengetahui bahwa suhu bervariasi 5 derajat sepanjang hari, dari 58 hingga 63 derajat sedikit berguna. Memiliki catatan menit demi menit biasanya tidak terlalu berguna.

Yang biasanya terjadi adalah semakin banyak informasi menghasilkan pertanyaan yang lebih baik, yang pada gilirannya menghasilkan lebih banyak pengumpulan informasi. Namun, ini tidak murah; Anda perlu meluangkan waktu untuk menerapkan intuisi dan penilaian pada analisis data ini, dan biasanya perlu menyewa sumber daya luar untuk menyiapkan dan menyetel sistem sehingga Anda mendapatkan informasi yang Anda butuhkan.

Industri lain, mulai dari manufaktur hingga minyak dan pertambangan, telah menemukan bahwa penyambungan seluruh rantai pasokan memungkinkan perencanaan yang lebih akurat, sehingga mengurangi kebutuhan akan kelebihan inventaris di lokasi. Dengan menerapkannya di semua material yang digunakan dalam proyek, akan terjadi peningkatan yang signifikan.

IoT benar-benar mengubah permainan karena memungkinkan manajer di semua tingkatan untuk menangani kompleksitas yang jauh lebih besar daripada yang seharusnya, karena apa yang dulunya hanya tebakan sebagian tentang seberapa cepat material digunakan, atau seberapa banyak material yang ada di lokasi, atau bagaimana cuaca memengaruhi pekerjaan, semuanya menjadi fakta nyata yang dapat dianalisis dengan pasti.

Cara terbaik untuk menjaga fokus manajer pada masalah yang sulit adalah dengan mengabaikan pengecekan status proyek secara terus-menerus yang dapat menghabiskan sebagian besar waktu dalam sehari. Inilah alasan utama mengapa dasbor sangat populer, dan juga mengapa kita melihat data sering dibuat grafik dan divisualisasikan: alat ini

memungkinkan referensi cepat ke metrik yang penting, dan dengan memvisualisasikannya, manajer dapat segera memahami perubahan dalam ukuran tersebut dan melihat posisinya terhadap tolok ukur.

8.4 IoT dan BIM

Model informasi bangunan (BIM) awalnya ditujukan untuk menggambarkan seperti apa sebuah bangunan dan bagaimana kinerjanya berdasarkan bahan, mesin, dan desainnya. IoT memungkinkan perusahaan untuk memperluas BIM dari gambar statis menjadi pemahaman dinamis tentang apa yang terjadi di bangunan mereka. Dengan tingkat hubungan antara desain, data yang dibangun, dan data pengoperasian ini, proses pembuatan, dan pengoperasian, sebuah bangunan dapat menjadi jauh lebih efisien.

BIM terkadang disebut sebagai "4D" atau "5D". Ini adalah permainan ide tentang 3D yang merupakan tiga dimensi, dengan "dimensi" ke-4 dan ke-5 adalah biaya dan jadwal. Dalam beberapa tahun terakhir, ini telah diperluas menjadi "6D", yang berarti menggunakan model informasi bangunan untuk menjalankan bangunan selama siklus hidupnya. Karena fase pengoperasian sebuah bangunan mewakili lebih dari 90% dari biaya seumur hidupnya, masuk akal untuk menghubungkan pengoperasian ini dengan representasi akurat bangunan dan aliran data berbasis sensor yang berkelanjutan tentang apa yang terjadi di bangunan tersebut. Konstruksi secara tradisional tidak terlalu mempedulikan apa yang terjadi setelah bangunan diserahkan. Tugas untuk memastikan bahwa apa yang dibangun memenuhi persyaratan operasional pemilik adalah tugas tim desain dan pemilik itu sendiri. Namun, banyak pemilik baru yang mengharuskan agar fase desain, pembangunan, dan pengoperasian diintegrasikan lebih erat, terutama perusahaan teknologi dan pengembang real estat skala besar seperti Proctor & Gamble. Di sinilah DFMA tumpang tindih dengan teknologi seperti IoT dan BIM, karena keputusan yang dibuat di awal tentang cara membangun desain bangunan dapat berdampak besar pada biaya operasional selama siklus hidup bangunan selama beberapa dekade – hal yang agak jelas yang mulai tercermin dalam sisi informasi dari proses tersebut.

Industri lain telah menemukan hubungan erat antara apa yang dibangun dan data operasional yang menciptakan peluang pendapatan yang besar. Mengumpulkan data, menganalisisnya, dan menyediakannya kembali kepada pemilik mesin jet, pembangkit listrik, dan lokomotif telah menjadi layanan berbayar bagi perusahaan seperti Siemens, GE, dan ABB, di antara banyak lainnya. Layanan ini memberikan nilai terbesar saat membuat prediksi kebutuhan pemeliharaan dalam apa yang disebut “kembaran digital”.

Kembaran Digital

Pada tahun 2000-an, perusahaan-perusahaan industri besar ini mulai memasarkan layanan data dan prediksi ini, dan istilah "kembaran digital" pun dicetuskan. Jika Anda membayangkan model BIM yang terhubung ke data operasi nyata, model tersebut akan mulai tampak sangat mirip dengan mesin atau bangunan dunia nyata yang diwakilinya. Jika pengaturannya benar, dan terdapat sensor di tempat yang tepat, kembaran digital akan menunjukkan panas, gerakan, dan keausan seperti aslinya, sehingga memungkinkan pemantauan jarak jauh terhadap benda nyata tersebut. Ide kembaran digital menjadi

kenyataan di semakin banyak mesin, pabrik industri, dan tentu saja pusat data. Ini merupakan perkembangan alami, dari bangunan buta di mana apa pun yang ingin Anda ketahui mengharuskan seorang pria paruh baya dengan seragam perawatan biru tua pudar untuk keluar dan melihat data operasi utama yang dibuat sebuah bangunan secara real-time, fleksibel, dan semakin lengkap. Sensor pada pipa, HVAC, dan elemen bangunan lain yang mungkin rusak akan menyediakan data yang dapat digunakan platform analitik untuk mengirim kru keluar sebelum kerusakan itu terjadi, atau setidaknya segera setelah kerusakan itu terjadi.

Salah satu hambatan untuk menginstrumentasikan bangunan nyata dengan cara yang sama seperti mesin adalah kesulitan melakukannya setelah bangunan dibangun. Kesulitan lainnya adalah bahwa sebagian besar sensor mengumpulkan satu atau dua jenis data, tetapi bagian tertentu dari bangunan dapat gagal dalam banyak hal – yang sering kali merupakan kesalahan penghuninya, bukan perancangnya. Karena ini adalah rentang titik kegagalan yang jauh lebih luas, didorong oleh rentang yang jauh lebih luas dari apa yang dilakukan orang di dalam bangunan, memilih sensor yang mungkin dapat mendeteksi masalah, dan dengan demikian menghasilkan keuntungan, jauh lebih sulit di dalam bangunan daripada di mesin jet.

Di sinilah visi mesin akan benar-benar mengubah cara kita menginstrumentasikan bangunan. Sebagai contoh, pada tahun 2016 kami menyelenggarakan acara teknologi di wilayah New York – hackathon selama akhir pekan dengan beberapa perusahaan IoT terkemuka di wilayah tersebut. Ini hanya empat tahun sebelum penerbitan buku ini, dan meskipun demikian kami dibingungkan oleh masalah yang jauh lebih mudah dengan layanan visi mesin yang tersedia saat ini. Secara khusus, menghitung orang di trotoar, mengukur seberapa dekat pejalan kaki dengan jalan, dan sebagainya, sangat sulit dilakukan dengan sensor jarak dan sensor berbasis gerakan lainnya. Sebaliknya, kamera apa pun cukup baik untuk merasakan hampir semua hal yang terlihat, karena visi mesin, ini adalah masalah perangkat lunak, bukan masalah perangkat keras. Dan masalah perangkat lunak selalu lebih baik, karena kita dapat mengubah perangkat lunak dengan mudah.

8.5 TEKNOLOGI YANG AKAN DATANG

Perusahaan seperti DPR, Mortensen, dan banyak lainnya memiliki tim inovasi khusus yang terus mencari cara untuk menggabungkan, dan menciptakan, teknologi dan aplikasi yang membuat operasi mereka lebih aman dan efisien. Dalam banyak kasus ini, mereka melakukannya karena pelanggan mereka mengharapkan inovasi, dan akan membayarnya. Ini jelas bukan kasus bagi banyak pemilik, jadi kemajuannya tidak merata.

Salah satu tujuan utama buku ini, dan bab ini, adalah untuk menurunkan risiko yang dirasakan dalam teknologi ini dan teknologi lainnya dengan menjelaskan apa itu dan apa fungsinya. Hampir semua teknologi yang akan penting untuk konstruksi memiliki versi tingkat konsumen yang dapat dibeli dan mulai digunakan oleh siapa pun. Meskipun membuat aplikasi Anda sendiri mungkin berada di luar jangkauan, dan biasanya di luar minat, sebagian besar kontraktor, ada cukup banyak bantuan di luar sana dalam bentuk serikat pekerja, inkubator,

dan analis sehingga tanpa terlalu banyak pekerjaan, Anda dapat mulai mencoba teknologi ini dan teknologi lainnya.

BAB 9
INOVASI DAN PENERAPAN TEKNOLOGI

"Penemuan adalah 1% inspirasi dan 99% kerja keras."
– Thomas Edison

Seperti banyak industri sebelumnya, konstruksi sedang mengalami transformasi digital besar-besaran. Hal ini terjadi karena tekanan di dalam dan luar industri konstruksi. Tekanan internal jelas terlihat – untuk produktivitas yang lebih tinggi, keselamatan yang lebih baik, dan kecepatan. Tekanan eksternal berkisar dari jenis pemilik baru, inovasi teknologi yang tak ada habisnya, dan persyaratan yang tidak pasti yang akan diberlakukan dunia pasca-Covid-19 kepada pemilik, arsitek, dan pembangun.

Transformasi digital menjanjikan untuk mengatasi tekanan ini dan dengan demikian memajukan konstruksi sebagai sebuah industri. Proses transformasi konstruksi menjadi industri yang canggih secara digital telah berjalan cukup jauh sehingga bagi sebagian besar perusahaan, mengadopsi teknologi baru merupakan kebutuhan yang kompetitif.

Hal yang sama dapat dikatakan untuk para profesional di lapangan. Baik sebagai pekerja khusus, pengawas, atau pekerjaan lain, perangkat digital menjadi bagian dari perangkat yang dibutuhkan. Dari bawah ke atas, konstruksi sedang mengalami transformasi. Proses transformasi dipandang sebagai bagian dari inovasi, dan kelompok yang bertugas mengelola penerimaan dan penerapan teknologi baru biasanya disebut kelompok "inovasi". Inovasi adalah kata yang besar, jadi kita akan menguraikannya dalam beberapa cara dan membangun definisi yang bermakna yang membantu Anda menyelesaikan pekerjaan Anda baik sekarang maupun di masa mendatang.

Ini adalah bab tentang tiga jenis inovasi yang akan membantu Anda mengambil bagian dalam dan mendorong transformasi digital:

1. Inovasi produk
2. Inovasi proses atau penggunaan internal
3. Inovasi untuk mengubah sikap

Pertama-tama, apa yang kita maksud dengan "inovasi?"

Inovasi adalah kreativitas yang diterapkan pada suatu produk atau layanan untuk menciptakan nilai yang lebih besar bagi bisnis atau pengguna.

Inovasi itu sendiri bukanlah kreativitas. Keduanya bercampur aduk, tetapi ada nilai dalam memisahkannya, karena masing-masing memerlukan proses yang berbeda.

Inovasi bukanlah ide, wawasan, atau "solusi." Inovasi adalah proses penerapan dan adaptasi ide untuk penggunaan nyata di dunia. Tidak ada ide yang siap digunakan begitu saja, masih banyak pekerjaan yang harus dilakukan untuk menyempurnakannya, dan mengembangkannya dengan pengguna nyata di lingkungan nyata. Karena begitu banyak

teknologi konstruksi yang harus dibuat di kantor tetapi digunakan di lapangan, bagian evolusi dari inovasi menjadi sangat penting bagi kami.

Sekarang setelah kita mengetahui apa itu inovasi, mari kita tinjau kembali tiga jenis inovasi yang mendorong transformasi digital: produk baru, proses baru, dan praktik perusahaan. Masing-masing layak mendapat bagiannya sendiri, dan kita akan mulai dengan jenis inovasi yang mungkin paling mudah dijelaskan – menciptakan produk baru.

9.1 INOVASI PRODUK

Inovasi produk dapat terjadi dalam beberapa cara: Vendor berinovasi dengan produk baru, pengguna menciptakan penggunaan baru untuk produk yang sudah ada, dan perusahaan secara internal mengembangkan produk atau konfigurasi produk yang sudah ada.

Inovasi Vendor

Kami telah membahas inovasi dan produk di seluruh buku ini – ada baiknya meninjau bagaimana profesional konstruksi dapat memahami apa yang ada di luar sana dan mengikuti perkembangannya tanpa membuang terlalu banyak waktu. Konstruksi adalah industri yang sangat personal. Orang-orang suka berbicara dengan orang lain, dan ini mengarah pada banyaknya konferensi, pertemuan, dan tempat lain tempat Anda dapat mempelajari apa yang ada di luar sana. Banyak dari tempat-tempat ini akan mengembangkan model hibrida tatap muka/virtual yang kemungkinan akan terus disesuaikan di tahun-tahun mendatang, yang berarti Anda dapat belajar banyak tanpa bepergian. Berikut ini adalah daftar yang bagus untuk membantu Anda memulai:

Daftar dan Blog

- Blue Collar Labs – ini adalah daftar perusahaan rintisan yang mencakup buletin yang menampilkan teknologi yang sedang naik daun (www.bluecollarlabs.com)
- JB Knowledge dan ConTech Crew – JBKnowledge memiliki laporan tahunan tentang teknologi, video, dan podcast mingguan yang disebut ConTech Crew yang merupakan cara yang bagus untuk mendengar dari perusahaan
- Capterra umumnya merupakan sumber informasi yang bagus untuk teknologi yang ada
- Construction Dive memiliki aliran artikel hebat tentang teknologi yang konstan
- ENR adalah yang terbesar, meliput seluruh industri, termasuk teknologi

Acara Serikat Pekerja

Semua orang dari MCAA, UA, AGC, TAUC, dan lainnya memiliki acara lokal dan nasional. Terlalu banyak untuk dicantumkan, dan banyak yang tertutup bagi anggota non-serikat pekerja, tetapi tetap layak untuk diperhatikan.

Acara dan Blog

- Autodesk University (AU) adalah pameran terbesar di AEC. Sedikit lebih fokus pada sisi desain, AU telah berkembang lebih fokus pada bidang dalam beberapa tahun terakhir.
- Procore Groundbreak semakin penting sebagai tempat untuk
- mempelajari Procore, tetapi juga banyak perusahaan mitra yang bekerja dengan Procore.
- ENR FutureTech adalah perluasan luring dari publikasi utama industri.

- ConTech Crew, perluasan luring dari podcast populer, adalah acara yang lebih seperti tur, dengan setidaknya mengunjungi kota-kota besar.

Acara hampir selalu mengenakan biaya masuk, tetapi blog dan videonya gratis. Berdasarkan pengalaman saya dengan semua ini, mereka selalu ingin mendengar dari lapangan.

Inovasi yang Dipimpin Pengguna

Pengguna biasanya berinovasi dengan menemukan cara baru untuk menggunakan produk lama. Inilah cara teknologi konstruksi modern di-bootstrap di banyak perusahaan, karena personel lapangan sudah terbiasa dengan teknologi digital di setiap aspek kehidupan mereka, dan mulai menerapkan pengalaman dan rasa pemberdayaan digital itu saat menghadapi tantangan di lokasi kerja.

Jika Anda telah menggunakan MS Excel di perguruan tinggi dan untuk keuangan Anda sendiri – atau sekadar terbiasa dengan kesederhanaan Netflix dan Amazon untuk hiburan – ide menjalankan semuanya melalui kertas tampak gila 10 tahun yang lalu, terlebih lagi sekarang. Sebelumnya saya merujuk pada komentar James Benham tentang konsumerisasi teknologi, dan inilah proses yang mengarah pada inovasi yang di-bootstrap yang telah berlangsung dalam konstruksi setidaknya selama satu dekade.

Misalnya, Blake Berg, pendiri DB, menceritakan kisah saat ia menjadi pengawas di perusahaan konstruksi ENR 50. Ia berada di kru yang sedang melakukan perbaikan pusat data yang luar biasa rumit, di mana tiga lantai server perlu dipindahkan ke satu lantai, semuanya tanpa mengganggu operasi inti pusat data.

Ini berarti bahwa ada aliran perubahan desain yang konstan, terutama pada pemasangan kabel dan penempatan mesin. Untuk melacak perubahan ini, dan untuk memberi kru yang sedang bekerja akses langsung ke gambar terbaru dan terkini, Blake menciptakan "aplikasi" layar sentuh yang berpusat pada folder yang berfungsi di tablet mana pun. Ini bukanlah lompatan teknis yang besar, ini hanya konfigurasi dan penerapan teknologi yang ada secara cerdas untuk memecahkan masalah.

Meskipun kata "inovasi" bernuansa teknologi, kenyataannya adalah bahwa setiap kru di setiap pekerjaan memecahkan masalah secara kreatif di lokasi kerja setiap hari. Ini sering kali merupakan masalah yang lebih kecil dan terjadi sekali saja, tetapi dasarnya sangat mirip.

Yang membuat seorang profesional perdagangan yang baik adalah beberapa pemecahan masalah yang sama yang membuat seorang inovator yang baik. Teknologi kini berarti bahwa pola pikir yang sama yang memungkinkan seorang tukang pipa untuk mencari tahu cara memecahkan salah satu dari banyak masalah yang muncul pada pekerjaan normal akan memungkinkannya untuk berinovasi secara lebih luas.

Ide inovasi yang dipimpin pengguna ini memiliki sejarah panjang, dan penting bagi ketiga pelaku dalam proses inovasi untuk memahaminya – pengguna harus tahu bahwa mereka adalah bagian dari tradisi yang kaya yang jauh melampaui konstruksi saat mereka mencoba membuat produk untuk melakukan hal-hal di luar apa yang dimaksudkan oleh pembuatnya. Perusahaan perlu mengetahui bagaimana pengguna mereka sebenarnya menggunakan produk tersebut, baik karena mereka harus tahu apakah mereka mendapatkan nilai, dan mungkin karena perusahaan dapat memilih untuk mendukung atau melarang

penggunaan baru ini jika itu berharga, atau dalam kasus yang jarang terjadi berbahaya. Seseorang mungkin memahami bagaimana teknologi dapat digunakan secara berbahaya, selain dari penggunaan mesin yang mungkin sembrono, tetapi bahaya yang tak terlihat dalam segala hal yang digital adalah bahwa hal itu menciptakan kerentanan keamanan yang dapat memengaruhi perusahaan dan pelanggan mereka.

Terakhir, vendor perangkat lunak dan teknologi itu sendiri perlu mengetahui bahwa pengguna di lapangan mungkin menggunakan produk mereka dengan cara baru. Bentuk inovasi ini telah dipelajari setidaknya sejak tahun 1980-an oleh para peneliti seperti Eric Von Hippel, yang menemukan bahwa banyak teknologi awal diperkenalkan tanpa pemahaman mendalam tentang bagaimana teknologi tersebut akan digunakan. Para pengadopsi awal kemudian mengambil produk dan membuat perubahan pada produk tersebut yang lebih sesuai dengan kondisi nyata.

Dalam industri seperti konstruksi, proses di lapangan bisa sangat rumit dan sulit dipahami dari luar. Pekerjaan aktual yang dilakukan di lapangan menurut definisinya merupakan adaptasi ide orang lain terhadap realitas situasi – bagaimanapun juga, menyiapkan pekerjaan berarti mencari tahu cara membuat rencana untuk membangun. Para pekerja dan kru GC mengadaptasi berbagai hal agar berfungsi sebagai keterampilan utama yang dibutuhkan – jadi tidak mengherankan jika teknologi diadaptasi agar berfungsi.

Nilai dari "mencari tahu" ini sering kali diabaikan oleh vendor teknologi yang berfokus pada proses internal mereka sendiri dan kompleksitas dalam menjual solusi kepada banyak pelanggan yang, selain adaptasi lapangan yang mungkin terjadi, memiliki perubahan yang didorong oleh kantor mereka sendiri. Satu perusahaan yang telah berupaya mendengarkan apa yang sebenarnya terjadi di lapangan adalah DADO, perusahaan perangkat lunak berusia dua tahun yang menemukan bahwa langkah pertama mereka sebagai perusahaan adalah keluar dari kantor dan menyelenggarakan, dengan cara apa pun yang mereka bisa, acara minum-minum, wawancara, dan diskusi dengan kru di lapangan. Apa yang mereka temukan mengejutkan mereka, dan benar-benar mengubah apa yang mereka pikir akan mereka lakukan dengan produk mereka.

Jake Olsen, CEO dan salah satu pendiri DADO, memberi tahu saya bagaimana mereka awalnya bermaksud membuat solusi yang berfokus pada BIM, tetapi setelah berbicara dengan orang-orang di lapangan, mereka menyadari bahwa akses yang cepat, mudah, dan akurat ke informasi apa pun yang mereka butuhkan adalah mata rantai yang hilang di lokasi kerja saat ini. Yang dibutuhkan oleh pekerja lapangan adalah semacam Google untuk lokasi kerja, dan DADO mulai membangunnya. Namun, mereka tidak berhenti di situ – DADO terus menyelenggarakan happy hour dan acara kecil lainnya bagi tim perdagangan untuk terus memahami cara penggunaan produk mereka, dan untuk menguji fitur-fitur baru. Disiplin ini langka dan sekaligus penting. Tidak cukup hanya melihat kebutuhan di pasar, kebutuhan tersebut harus terus divalidasi. Inovasi bukan hanya tentang ide-ide baru, tetapi terutama tentang jutaan keputusan yang dibuat untuk mengubah suatu produk menjadi kenyataan yang berhasil di pasar, dan kerja sama erat DADO dengan pekerja lapangan memastikan bahwa inovasi mereka melayani pengguna yang dituju dengan jauh lebih efektif.

Seperti yang disebutkan Jake: “Sebagian besar pekerja lapangan tidak tahu betapa hausnya para teknolog akan wawasan mereka. Kami perlu mengetahui apa yang mereka butuhkan sehingga kami dapat membuat hal-hal yang akan membantu.”

Bagi Anda yang bekerja di lapangan, saya berharap dapat mendengar dari Anda; kami membutuhkan suara Anda untuk membantu mendorong inovasi yang lebih baik.

9.2 INOVASI INTERNAL

Banyak perusahaan memiliki tim teknologi mereka sendiri, sering kali hanya beberapa pengembang yang membangun koneksi antara berbagai jenis perangkat lunak, tetapi terkadang lebih. Seberapa banyak lagi tergantung pada strategi perusahaan dan seberapa banyak upaya yang ingin mereka curahkan untuk menjadikan perangkat lunak dan teknologi lainnya sebagai bagian penting dari apa yang membuat mereka lebih baik daripada pesaing.

Salah satu cara untuk mengetahui apakah sebuah perusahaan memandang teknologi sebagai hal yang penting bagi daya saing mereka adalah jika mereka memiliki Chief Technology Officer (CTO). Bagi banyak orang, perbedaan antara Chief Innovation Officer (CIO) dan CTO tidaklah jelas – jadi mari kita perjelas.

Seorang CTO harus fokus pada teknologi yang ditawarkan perusahaan ke pasar, kepada pelanggan di luar perusahaan. Seorang CIO fokus pada teknologi yang digunakan dalam operasi inti perusahaan. Jadi, tim IT berada di bawah CIO, sedangkan tim produk berada di bawah CTO. Perusahaan konstruksi bukanlah perusahaan perangkat lunak, jadi memiliki CTO mungkin akan tetap menjadi hal yang langka, tetapi apa yang telah kita lihat dalam konstruksi, serta di tempat lain, adalah bahwa solusi perangkat lunak yang berbeda yang telah diadaptasi dan diperiksa oleh kru di dalam perusahaan terkadang dapat dijual di luar perusahaan. Secara pribadi, saya pikir perhatian manajerial yang benar-benar dibutuhkan untuk mengomersialkan perangkat lunak merupakan gangguan bagi sebagian besar perusahaan, tetapi itu memang terjadi. Saya bertanya kepada Pat Sharpe, seorang teknolog AEC seumur hidup, untuk beberapa contoh, dan berikut ini beberapa contoh yang menurutnya bagus:

- Bovis Lend Lease: Hummingbird (Manajemen Gambar)
- Arup: Columbus (Navigasi Berkas)
- Leighton: Incite (Manajemen Proyek) (diakuisisi oleh Aconex yang kemudian diakuisisi oleh Oracle)
- T&G Constructors: RedTeam Software (Back Office)
- Katerra: Apollo (Rangkaian Perangkat Lunak Konstruksi Modular)

Namun, lebih sering, inovasi internal adalah tentang menghubungkan produk yang sudah ada sehingga produk tersebut bekerja sesuai dengan keinginan perusahaan, atau sekadar menghubungkannya. Sudah lama menjadi masalah, baik di dalam maupun di luar perangkat lunak konstruksi, bahwa satu perangkat lunak tidak bekerja dengan perangkat lunak lain, meskipun apa yang mereka lakukan saling terkait erat – misalnya, perangkat lunak manajemen konstruksi yang tidak terhubung dengan solusi estimasi, atau dengan sistem akuntansi. Hal ini juga dapat terjadi dengan jenis dan format berkas. Misalnya, tim desain sering kali menyediakan model BIM berbasis Revit yang tidak sesuai untuk mesin CNC, jadi perlu ada

sesuatu yang menerjemahkannya. Ada beberapa opsi komersial, tetapi beberapa perusahaan telah menemukan solusi mereka sendiri.

Inovasi internal sangat sering kali lebih erat kaitannya dengan proses khusus perusahaan dan dapat menjadi sumber keunggulan kompetitif yang besar, setidaknya karena dua alasan. Yang pertama adalah bahwa apa pun yang Anda bangun akan menjadi perluasan dari proses yang telah Anda inovasi, dan itu hampir mustahil untuk ditiru. Yang kedua adalah teknologi itu sendiri tidak mungkin dikomersialkan, dan juga sangat sulit untuk ditiru dan dibuat berguna di luar teknologi spesifik yang digunakan perusahaan.

Bagian yang sulit untuk inovasi internal adalah bahwa perusahaan konstruksi bukanlah perusahaan perangkat lunak, jadi pekerja rata-rata tidak menganggap teknologi sebagai solusi – dan jika mereka menganggapnya demikian, sangat sering merasa tidak mampu melakukan apa pun tentang hal itu. Seperti yang akan kita lihat lebih rinci, jawaban untuk ini adalah dengan mengubah budaya perusahaan dari pendekatan top-down yang berfokus pada pelaksanaan yang merupakan norma selama bertahun-tahun, menjadi budaya yang mendorong ide-ide dan memiliki proses untuk benar-benar mewujudkan beberapa ide ini menjadi kenyataan.

Pergeseran ke pendekatan bottom-up ini dilakukan dengan berbagai cara, mulai dari kontes inovasi hingga hackathon hingga sesi design thinking. Semuanya membantu, bukan karena setiap aktivitas individu seperti kontes atau hackathon akan menghasilkan ide yang unggul, tetapi karena seiring waktu mereka mengubah budaya ke arah budaya yang merangkul teknologi sebagai alat yang dapat mereka ciptakan, yang dapat mereka miliki, alih-alih perangkat lunak yang tidak peka terhadap nada yang sulit digunakan dan tidak benar-benar melakukan apa yang seharusnya dilakukan.

9.3 INOVASI SEBAGAI PERUBAHAN BUDAYA

Sering kali ada anggapan dalam konstruksi bahwa "kami selalu melakukannya dengan cara ini," sehingga proses baru memiliki sedikitnya, tetapi sering kali lebih banyak, hambatan terhadap inovasi daripada produk. Dan seperti yang Anda duga, banyak vendor teknologi mengungkapkan rasa frustrasi atas kurangnya keinginan untuk mencoba hal-hal baru.

Pertama, "selalu" biasanya berarti sekitar 25–30 tahun, atau lamanya waktu rata-rata profesional konstruksi telah berkecimpung di pasar. Generasi pekerja baru selalu menyebabkan perubahan di tempat kerja mereka, dan itu pasti terjadi dalam konstruksi.

Sebenarnya, sebelum kita menyelami inovasi berbasis proses, mari kita luangkan waktu sejenak untuk memikirkan seberapa terbuka atau bersemangatnya tenaga kerja di masa mendatang terhadap perubahan ini, karena dua alasan terkait: perubahan demografi dan perubahan yang disebabkan oleh krisis.

Perubahan Demografi

Telah dikatakan bahwa "demografi adalah takdir." Bagi kita yang berbicara tentang teknologi, manusialah yang membangun sebuah industri, dan manusialah yang menentukan karakter industri tersebut. Ketika generasi baru datang dan pergi, mereka membawa serta perubahan. Mari kita lihat apa artinya itu.

Menandai generasi yang berbeda bukanlah ilmu pengetahuan, karena kita memiliki generasi baru setiap tahun. Namun, nilai dari generasi yang dikelompokkan, seperti Baby Boomers, adalah bahwa mereka semua akan mengalami banyak hal yang sama dan dengan demikian memiliki pandangan hidup yang sama. Karena satu generasi menyiratkan satu siklus kelahiran, kedewasaan, dan kelahiran, generasi biasanya berlangsung selama 20 tahun, yang membuatnya kurang berguna sebagai cara untuk menggambarkan sekelompok orang, karena tentu saja beberapa dari apa yang merupakan pengalaman formatif bagi anggota generasi yang lebih tua dan lahir lebih awal akan terjadi sebelum anggota yang lebih muda dan lahir kemudian lahir. Misalnya, generasi Baby Boomer adalah mereka yang lahir dari tahun 1946 hingga 1964. Hal ini dikarenakan di AS dan tempat lain, para tentara pulang dari Perang Dunia II dan memiliki banyak bayi. Generasi-generasi tersebut secara umum dipahami sebagai berikut:

1946–1964: Baby Boomer
1965–1980: Generasi X
1981–2000: Milenial
2001–2020: Gen Z

Anda akan melihat bahwa Gen X sedikit lebih pendek dari yang lain – hal ini sebagian karena Gen X dicirikan oleh angka kelahiran yang rendah, yang mulai berubah sekitar tahun 1980. Akibatnya, Gen X adalah generasi terkecil baik karena angka kelahiran yang lebih rendah per tahun, maupun jumlah tahun yang lebih sedikit. Seperti yang saya katakan, hal ini bukanlah ilmu pengetahuan.

Generasi Boomer mendefinisikan keadaan industri konstruksi saat ini. Mereka bertanggung jawab untuk membawa industri yang mereka ikuti pada tahun 1970-an dan 80-an dan mengembangkannya dengan teknologi dan lingkungan hukum saat itu menjadi apa yang kita lihat saat ini. Generasi X, yang sudah merupakan generasi kecil, tidak memiliki insentif untuk tetap bekerja di bidang konstruksi selama krisis keuangan 2008, yang menyebabkan kekurangan bakat yang serius. Mengingat bahwa generasi X termuda berusia 40 tahun saat artikel ini ditulis, kecil kemungkinan mereka akan menjadi bagian dari penyelesaian kekurangan bakat tersebut. Generasi milenial mengalami krisis tahun 2000, 9/11, tingkat utang mahasiswa yang jauh lebih besar dari orang tua mereka, krisis tahun 2008, dan sekarang semua krisis tahun 2020. Generasi ini adalah generasi yang paling beragam dari semua negara besar dalam sejarah. Sama seperti Generasi Baby Boomer, yang menuntut perubahan pada tahun 1960-an, Generasi Milenial menuntut perubahan dalam kehidupan kerja mereka, dan hal ini terlihat dari terbatasnya partisipasi mereka dalam konstruksi. Seperti yang banyak dari Anda ketahui, usia rata-rata karyawan konstruksi adalah 47 tahun, yang berarti bahwa generasi milenial belum mengambil tempat mereka dalam konstruksi, per tahun 2020. Sebelum pandemi, hal ini menyebabkan banyak perusahaan konstruksi mempertanyakan bagaimana mereka melakukan sesuatu, untuk melihat budaya dan proses yang selama ini mereka andalkan.

Sebagian besar fokusnya adalah pada kolaborasi dan budaya yang terkenal kasar – sesuatu yang umumnya tidak mau diterima oleh Generasi Milenial. Namun yang sama

pentingnya, Generasi Milenial tidak hanya mengharapkan, mereka membutuhkan tempat kerja digital.

Jadi, cerita demografisnya adalah bahwa Generasi Milenial akan membentuk sebagian besar industri konstruksi di masa mendatang, dan mereka sudah menuntut lokasi kerja yang lebih inklusif, tidak terlalu kasar, dan lebih terintegrasi secara teknologi. Sangat mungkin bahwa perusahaan yang menggabungkan fabrikasi di luar lokasi dan konstruksi industri secara umum akan menemukan bahwa mereka dapat menarik lebih banyak pekerja dengan biaya lebih rendah daripada yang tidak. Generasi Milenial akan mendorong perubahan yang sama besarnya seperti orang tua mereka yang generasi Boomer, dan itu akan dimulai dengan kebutuhan untuk mengubah proses dalam konstruksi.

Perubahan yang Didorong Krisis

Sebelum pandemi 2020, ada anggapan umum bahwa, agar efektif, tim perlu bekerja sama. Dan 20 tahun yang lalu, itu akan benar, karena telepon bukanlah cara yang bagus untuk mengomunikasikan ide-ide yang rumit dan cara yang buruk untuk berkolaborasi.

Pada paruh pertama tahun 2020, ekonomi dunia terkunci dan dalam beberapa hari sekitar dua pertiga ekonomi AS beralih dari tatap muka ke Zoom. Hal yang sama ini terjadi di seluruh dunia, dengan berbagai negara mengikuti lintasan yang sama pada waktu yang berbeda dan dengan cara mereka sendiri.

Tiba-tiba teknologi yang berguna tetapi tidak penting menjadi penting dalam cara kita melakukan sesuatu – beberapa dari kita praktis menggunakan Zoom, Google Hangouts, atau Microsoft Teams. Rapat yang sebelumnya harus dilakukan secara langsung disesuaikan menjadi virtual. Dan teknologi seperti Holobuilder, Openspace.ai, dan lainnya yang memungkinkan survei jarak jauh di lokasi kerja tiba-tiba menjadi jauh lebih menarik, dan tentu saja siapa pun dengan teknologi apa pun yang dapat membantu komunikasi jarak jauh menambahkan fitur dalam waktu singkat. Pada saat publikasi, krisis Covid-19 belum berakhir, karena vaksin maupun pengobatan yang efektif belum tersedia, tetapi bagaimanapun krisis berakhir, krisis berlangsung cukup lama untuk mengguncang sistem sehingga proses secara umum akan jauh lebih terbuka terhadap perubahan dan evolusi daripada yang terjadi di masa lalu.

Seiring berjalannya dekade 2020–2030, kita tidak dapat mengetahui dengan pasti apa yang akan terjadi atau proses apa yang akan berkembang. Yang dapat kita prediksi dengan yakin adalah bahwa industri konstruksi akan lebih terbuka terhadap pemikiran ulang yang radikal tentang bagaimana segala sesuatunya dilakukan daripada yang telah terjadi selama bertahun-tahun, jika tidak pernah terjadi sebelumnya. Ada badai yang sempurna dari persyaratan bangunan yang terus meningkat, tenaga kerja yang menyusut dan menuntut, dan asumsi yang sangat menantang tentang cara menyelesaikan pekerjaan yang digabungkan untuk menciptakan potensi perubahan nyata.

Inovasi Proses

Banyak sektor ekonomi yang proses intinya telah didesain ulang dalam beberapa tahun terakhir. Manufaktur, operasi rumah sakit, keuangan dan perbankan, periklanan – semuanya adalah daftar yang panjang. Semuanya telah berubah karena kehadiran teknologi dan

pergeseran mendasar di pasar mereka, termasuk meningkatnya kecanggihan di pihak manajer.

Faktanya, dari tahun 1980-an hingga akhir abad lalu, aliran ide yang stabil membanjiri lanskap bisnis, meninggalkan kantor, budaya, dan proses yang sangat berbeda. Meskipun perubahan seperti itu rumit, salah satu pendorong terbesar adalah fleksibilitas yang dibawa oleh digitalisasi. Di masa lalu, sebagian besar informasi terkunci di kertas, atau tidak benar-benar dikumpulkan sama sekali. Kemudian jenis perangkat lunak baru muncul, dan mengubah rantai pasokan, manufaktur, keuangan, dan SDM – yang disebut "Perencanaan Sumber Daya Perusahaan," dan dampaknya adalah membuat informasi bebas dari media apa pun. Data tersebut dapat ditabulasi, dianalisis, dan dikirim ke mana saja untuk digunakan dalam pengambilan keputusan tentang bagaimana segala sesuatunya diselesaikan.

Pada saat yang sama, dunia bisnis sedang pulih dari guncangan yang ditimbulkan ekonomi Jepang terhadap dunia barat – proses tidak hanya penting, tetapi juga dapat benar-benar berbeda dari apa yang telah dilakukan sebelumnya, dengan hasil yang hampir ajaib. Sistem Produksi Toyota (TPS), yang kemudian berkembang menjadi Lean manufacturing, Six Sigma, dan model rekayasa ulang proses lainnya, semuanya mampu memfokuskan kembali perusahaan dengan cara yang mendorong peningkatan produktivitas, dalam beberapa kasus secara radikal.

Dengan kata lain, teknologi telah hadir untuk membebaskan data dan informasi dari kertas dan interaksi antarpribadi yang telah menguncinya, sama seperti guncangan eksternal telah menciptakan kesadaran bahwa bekerja dengan cara lain adalah mungkin, dan sering kali lebih baik.

Kita tidak dapat mengetahui apa yang akan terjadi selanjutnya, karena industri konstruksi sangat tersebar sehingga perubahan sulit dilakukan, tetapi panggungnya sudah pasti siap untuk desain ulang proses, terutama di ENR 400.

> *Bagaimana kita mendesain ulang proses?*
> *Bagaimana kita berinovasi dengan cara-cara baru untuk mengatur orang, mengatur upaya mereka, dan mengoordinasikan aktivitas mereka di seluruh rantai pasokan?*

Mari kita mengambil jalan memutar dan melihat fondasi inovasi – proses berpikir yang dapat menghasilkan ide-ide yang, pada gilirannya, memberi kita inovasi.

9.4 INOVASI VERSUS KREATIVITAS

Perusahaan desain dan agensi periklanan telah menjual kemampuan mereka untuk menghasilkan ide selama seratus tahun. Ide-ide ini diperlukan untuk memecahkan masalah, yang sering kali didefinisikan dengan buruk. Dan perusahaan-perusahaan tersebut harus mampu menghasilkan serangkaian ide-ide bagus, karena klien memiliki kebiasaan buruk untuk tidak menyukai hal pertama yang Anda tunjukkan kepada mereka. Jadi, perusahaan-perusahaan kreatif ini mengembangkan metode yang telah dicoba dan diuji untuk menghasilkan ide-ide, untuk berpikir kreatif dan menghasilkan ide-ide yang dapat mengubah dunia.

Ide-ide bagus itu sulit. "Writers' block" tidak dapat diterima ketika tenggat waktu semakin dekat – jadi pada awal abad kedua puluh kreativitas dipelajari, sampai pada titik di mana kita mulai memahami cara kerjanya. Hal ini diperlukan karena mampu menentukan cara kerja sesuatu berarti Anda dapat merancang proses untuk mengerjakannya saat diperlukan. penulis memiliki definisi kreativitas yang jelas terkait dengan inovasi, tetapi lebih kepada ide, bukan aplikasi.

Kreativitas adalah kombinasi dari dua atau lebih ide dengan cara yang baru dan bermanfaat.

Sama seperti inovasi, ide perlu memberikan nilai tambah agar dianggap kreatif, tetapi ide utama di sini adalah penggabungan dua ide lain yang lebih kecil. Itu berarti Anda perlu memiliki sekumpulan ide yang lebih kecil dan lama di kepala Anda agar dapat menggabungkannya dengan bermanfaat. Bagian pertama dari kreativitas adalah kerja keras, belajar, berpikir, dan berdebat sampai Anda mengisi kepala Anda dengan fakta, bagian dari ide, dan sedikit pemahaman tentang cara kerja berbagai hal di dunia masalah Anda.

Masalahnya, otak kita tidak menyukai fakta yang campur aduk yang tidak cocok satu sama lain, jadi alam bawah sadar kita berusaha untuk mencocokkannya, untuk menemukan hubungan yang masuk akal dari semuanya. Kita dapat menganggap ide sebagai simpul dalam jaringan, yang terhubung dengan ide lain, dan semakin banyak waktu yang Anda habiskan untuk semua ide ini, memikirkannya, dan mencoba menghubungkannya, semakin Anda mencoba menyatukannya sehingga muncul solusi baru, dan semakin baik Anda mempersiapkan otak untuk membuat koneksi itu saat itu juga, saat Anda berpikir, atau nanti saat Anda tidak menduganya.

Dan bagi banyak orang, ide terbaik mereka muncul saat mereka tidak menduganya. Ini bukan hanya untuk periklanan atau desainer kreatif. Para peneliti telah mengamati bagaimana fisikawan, penulis, ahli strategi bisnis, dan insinyur memunculkan ide terbaik mereka, dan polanya sama. Ada lima langkah proses yang diterima secara umum untuk memunculkan ide-ide baru:

1. Tentukan masalahnya
2. Pelajari semua yang Anda bisa tentang subjek yang terlibat dalam masalah tersebut, dan berusahalah keras untuk memecahkan masalah tersebut
3. Jauhi masalah tersebut – tidurlah, jalan-jalan, apa pun
4. Kembalilah, dan cobalah untuk memecahkan masalah tersebut lagi
5. Sempurnakan ide

Ini terdengar seperti cara yang tidak pasti untuk melakukannya, terutama langkah #3. Namun, di situlah keajaiban terjadi. Otak kita cenderung terus mengerjakan masalah tersebut bahkan saat kita tidak secara aktif mencoba – faktanya, relaksasi pikiranlah yang memungkinkan koneksi baru dan tak terduga tercipta dan muncul ke permukaan.

Jika Anda pernah mendapat ide saat mandi, atau saat makan malam, atau menyadari bahwa wawasan baru sering kali muncul saat Anda sedang berlibur, itu semua prosesnya sama saja. Terkadang lebih baik beristirahat, minum bir, dan membicarakan hal lain agar otak Anda dapat bekerja. Kreativitas merupakan bagian penting dari inovasi, dan sebagai ahli dalam beberapa bagian dari proses konstruksi, setiap orang dalam pekerjaan tersebut memiliki

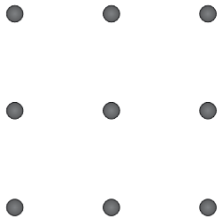
kapasitas untuk berinovasi dalam beberapa solusi baru, beberapa pendekatan baru terhadap masalah.

Berpikir Lateral

Cara berpikir yang terkait dengan kreativitas adalah apa yang disebut oleh Edward de Bono, seorang peneliti kreativitas, sebagai "berpikir lateral." Ketika Anda diminta untuk "berpikir di luar kotak," inilah yang mereka maksud.

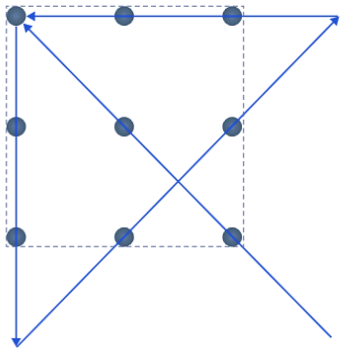
Berpikir lateral berarti melihat masalah dalam konteks yang lebih luas, untuk menemukan solusi yang tidak jelas jika Anda tetap menggunakan solusi dan pendekatan yang biasa Anda gunakan. Bahkan, mari kita gunakan contoh "di luar kotak," untuk mengilustrasikan apa arti berpikir lateral.

Frasa "berpikir di luar kotak" pada awalnya tidak ada hubungannya dengan kotak sebenarnya tempat Anda sebagai pemikir terjebak; ini merujuk pada teka-teki. Teka-teki ini sudah ada sejak lama, dan saya ingin Anda mencoba teka-teki tersebut sebelum membalik halaman untuk mendapatkan jawabannya. Teka-teki ini dikenal sebagai masalah sembilan titik, dan teka-teki ini sederhana:



Bagaimana Anda menghubungkan kesembilan titik dalam matriks ini hanya dengan empat goresan, dan tidak pernah mengangkat pena Anda?

Kebanyakan orang mencoba menggambar garis dari satu sudut ke sudut lain, atau sepanjang sisi, dan merasa tidak berhasil. Mereka terpaku pada ide persegi, atau "kotak." Untuk mengatasinya, penting untuk mengabaikan pola titik-titik yang jelas sebagai "kotak" dan hanya menganggapnya sebagai titik-titik yang harus dihubungkan. Bagaimana jika garis yang menghubungkan titik-titik ini memanjang ke luar "kotak?" Maka akan terlihat seperti ini:



Berpikir lateral, dengan demikian, memperluas masalah sehingga Anda dapat menemukan sumber daya lain yang dapat membantu Anda menyelesaikannya. Contoh ini menunjukkan kepada kita bahwa yang menghalangi kita untuk berpikir lateral, dari berpikir di luar kotak, sering kali adalah asumsi yang kita buat tentang bagaimana segala sesuatu seharusnya bekerja. Kita membuat asumsi tentang apa yang mungkin, diizinkan, atau paling efektif, dan sering kali asumsi tersebut ternyata salah. Terutama dalam industri di mana teknologi terus berubah, asumsi tentang apa yang dapat dilakukan menjadi usang dengan cepat.

Berpikir lateral sebagai sebuah ide dapat membantu kita menyadari bahwa kita mungkin berpikir terlalu sempit dan perlu membuka apa yang sedang kita pertimbangkan, dengan menerobos asumsi kita.

Jadi, bagaimana kita mematahkan asumsi ini? Mungkin yang terbaik di dunia dalam mempertanyakan asumsi dan berpikir di luar kotak adalah

Elon Musk.

Ia terkenal karena mendefinisikan ulang berbagai industri, dari pembayaran hingga mobil listrik hingga pesawat ruang angkasa dan banyak lagi. Musk melakukan ini melalui komitmen radikal terhadap pemikiran prinsip pertama, yang merupakan cara Anda berdua melewati asumsi dan kemudian membangun cara berpikir baru setelah Anda melakukannya.

Pemikiran Prinsip Pertama

Kebanyakan orang hanya berpikir dengan analogi. Berpikir dengan analogi berarti Anda melihat masalah atau situasi Anda, kemudian mencari contoh dalam ingatan yang tampak seperti itu. Anda kemudian mencari solusi yang berhasil untuk situasi serupa ini. Anda memilih solusi yang tampaknya paling mungkin berhasil, kemudian mensimulasikan solusi itu dalam pikiran Anda. Jika berhasil, Anda selesai – Anda memiliki solusi Anda. Jika tidak, Anda beralih ke kemungkinan lain dan mencobanya. Jika tampaknya akan berhasil, sekarang Anda memiliki solusi Anda, dan jika tidak, ulangi saja prosesnya.

Penalaran dengan analogi adalah yang menjadikan Anda seorang ahli, yang memungkinkan Anda melakukan sesuatu dengan cepat, melakukannya dengan baik, dan menemukan kesalahan dengan cepat. Anda seharusnya menghabiskan 99% hidup Anda untuk bernalar dengan analogi; orang yang tidak berpikir dengan analogi tidak berpengalaman dan mencoba mencari tahu sesuatu dari apa yang mereka lihat. Ini adalah cara yang buruk untuk melakukan sesuatu yang sudah diketahui seseorang, dan itulah mengapa pelatihan sangat penting.

Sebaliknya, penalaran dengan prinsip-prinsip dasar itu sulit, mungkin paling baik dilakukan hanya ketika Anda mencoba mencari tahu sesuatu yang penting, dan paling baik dilakukan dalam kelompok. Prosesnya dapat dianggap sebagai perjuangan Anda menuruni tangga asumsi, kemudian membangun rencana atau proses Anda dari dasar.

Ada empat tahap pemikiran prinsip-prinsip dasar: tentukan tujuan Anda; hilangkan asumsi; tentukan prinsip-prinsip dasar Anda; kemudian gunakan logika "jika, maka" untuk membangun solusi. Proses ini terlihat seperti ini:

1. ***Tentukan tujuan Anda***. Ini terdengar mudah, tetapi sebagian besar waktu, pernyataan pertama tujuan Anda mencakup beberapa cara tersirat untuk memecahkan masalah. Faktanya, sebagian besar masalah didefinisikan terlalu dekat dengan solusinya, padahal yang sebenarnya Anda selesaikan sebenarnya adalah tujuan tingkat tinggi.
   Misalnya, "kita perlu menyelesaikan lebih dari 90% laporan harian kita." Laporan harian sepertinya bukan tujuan yang baik, karena tidak seorang pun benar-benar membutuhkan laporan harian. Anda memerlukan data yang dikumpulkan laporan harian, dan Anda harus dapat membuktikan bahwa data tersebut valid, bahwa data tersebut mencerminkan realitas yang ingin direpresentasikannya. Mari kita nyatakan kembali tujuannya sebagai "kita perlu mendapatkan data dari 90% atau lebih pekerja di tempat kita tentang kapan mereka berada di sana, apa yang mereka lakukan, dan masalah apa pun yang muncul."
2. ***Hilangkan asumsi.*** Ini tidak mudah, dan sekali lagi mungkin harus dilakukan sebagai sebuah tim, karena orang-orang pada umumnya lebih baik dalam menemukan

kesalahan dalam pemikiran orang lain daripada menyadari jalan pintas dan asumsi bawaan mereka sendiri. Memperluas sasaran laporan harian, mari kita lihat apa yang mungkin diasumsikan di sana:

- Anda perlu mengumpulkan ini dari mereka
- Anda perlu menyajikan ini untuk gugatan hukum di masa mendatang
- Anda hanya memerlukan informasi ini
- Anda hanya akan menggunakan informasi ini untuk perselisihan di masa mendatang
- Informasi ini perlu dikumpulkan sebagai latihan terpisah
- Informasi ini perlu dikumpulkan secara aktif oleh mandor dan pengawas
- Informasi akan 100% akurat, dan tidak berguna jika tidak 100% akurat.

3. ***Tetapkan prinsip-prinsip pertama.*** Setelah Anda menyingkirkan sebanyak mungkin asumsi, apa yang tersisa? Apakah itu cukup untuk menyelesaikan masalah Anda? Sering kali, prinsip-prinsip yang disederhanakan yang tersisa membuka kemungkinan baru, arena baru yang mungkin Anda sertakan dalam solusi Anda. Misalnya, bagaimana jika masalah Anda sekarang tampak seperti ini:

   "Kita perlu membuat daftar yang berkelanjutan dan real-time tentang siapa yang ada di lokasi, di mana mereka berada, dan apa yang mereka lakukan yang setidaknya x% lengkap dan x% akurat."

   Bagaimana saya sampai pada hal itu? Pikirkan tentang apa yang diasumsikan oleh daftar di akhir hari tentang siapa yang ada di sana – yang mengasumsikan Anda tidak dapat memahami setiap menit. Yang mengasumsikan Anda tidak akan membutuhkan data yang lebih baik. Dan yang mengasumsikan Anda tidak dapat melakukan yang lebih baik. Yang mengasumsikan juga bahwa Anda tidak dapat bekerja dengan data yang bersifat probabilitas. Hal-hal ini tidak perlu benar.
4. ***Bangun solusi***. Dengan membuka masalah kita hingga ke akar-akarnya, kita dapat mulai bertanya tentang cara untuk mengetahui apa yang sedang terjadi pada interval yang berbeda, atau terus-menerus. Bisakah kita memasang pelacak pada helm pengaman? Itu sudah dilakukan. Bisakah kita memasang kamera yang menggunakan pengenalan wajah? Itu sudah dilakukan. Jika kita menggunakan pengenalan wajah, tetapi terkadang gelap, atau Anda tidak dapat memastikan siapa yang berada di suatu tempat, apakah itu masih baik-baik saja?
   Bagaimana dengan privasi? Bagaimana dengan jangkauan kamera?
   Solusi untuk ini mungkin memasang kamera 360^{o} sederhana di lokasi kerja utama, dan kamera video standar di pintu masuk, memanfaatkan AI dan membangun pemahaman lengkap tentang tim mana yang berada di mana, dan menghubungkannya dengan persentase penyelesaian. Atau mungkin sesuatu yang lain; intinya adalah bahwa dengan menghilangkan asumsi, kita dapat berpikir secara lateral dan out of the box, dari prinsip-prinsip awal.

Pemikiran prinsip-prinsip awal adalah cara yang ampuh untuk melihat proses dengan cara baru, mengidentifikasi asumsi yang membatasi, dan menciptakan solusi baru. Jika ada kelemahan, pemikiran prinsip pertama tidak benar-benar memberi kita cara untuk benar-benar memecahkan masalah. Fokusnya adalah menyingkirkan asumsi, tetapi kita memerlukan beberapa metodologi untuk membangun solusi apa pun yang kita butuhkan.

Dalam kasus inovasi proses, kita akan bekerja dengan cara orang berinteraksi satu sama lain, cara mereka menggunakan alat bersama, dan apa yang orang pikirkan tentang cara-cara yang ada dalam melakukan sesuatu. Alat terbaik untuk masalah semacam ini adalah pemikiran desain, yang akan kita bahas selanjutnya.

9.5 PEMIKIRAN DESAIN

Seiring waktu, desainer dan inovator telah memperluas proses kreativitas, dan mengembangkan apa yang kita kenal sebagai "pemikiran desain." Pemikiran desain mengikuti beberapa ide dasar yang sama dengan pemikiran prinsip pertama, karena fokusnya terutama pada menghindari asumsi saat Anda membuat solusi - dalam hal ini, asumsi tersebut adalah apa yang terjadi ketika ide hanya berupa omongan dan gambar. Asumsinya adalah bahwa orang-orang yang mengerjakan masalah tersebut semuanya memahami apa arti omongan dan gambar tersebut, padahal kita tahu bahwa biasanya mereka tidak memahaminya. Dalam pemikiran desain, tim menciptakan prototipe konkret dan mengulanginya dengan umpan balik pengguna yang nyata sejak dini dan sering. Dalam setiap kasus, kami memperoleh informasi yang jelas dan nyata, bukan asumsi dan generalisasi, dan itu mengarah pada solusi yang lebih baik.

Langkah-langkah pemikiran desain adalah: berempati, mendefinisikan, mengideasi, membuat prototipe, dan menguji. Langkah-langkah khusus ini berasal dari perusahaan-perusahaan yang telah menciptakan solusi selama bertahun-tahun yang sebenarnya tidak bekerja untuk orang-orang. Kegagalan tersebut menghasilkan pelajaran penting tentang cara memulai dengan cara yang benar, menjaga pengembangan tetap pada jalurnya, dan memanfaatkan sebaik-baiknya cara manusia berpikir di luar kotak.

1. ***Berempati:*** Kedengarannya jelas, tetapi untuk membangun sesuatu yang bermanfaat, Anda perlu memahami masalah dan bagaimana solusi Anda akan benar-benar digunakan. Ini berarti lebih dari sekadar memahami fungsi apa yang perlu dilakukan, ini berarti memahami siapa yang akan melakukannya, dan bagaimana mereka akan melakukannya.

   Langkah ini tumpang tindih dengan Pemikiran Prinsip Pertama. Proses ini sering difasilitasi oleh seorang ahli yang akan mengajukan pertanyaan sulit tentang asumsi yang dibuat saat data dikumpulkan tentang kebutuhan pelanggan yang sebenarnya. Paling tidak yang terpenting, moderator dan tim secara keseluruhan perlu bersikeras pada masukan pengguna yang sebenarnya, bukan hanya curah pendapat tentang apa yang mungkin mereka butuhkan.

   Kami menyebut langkah ini "berempati" karena ini tentang memahami pengguna. Tidak banyak industri di mana kesenjangan antara pengetahuan kantor dan

pemahaman lapangan bisa selebar konstruksi, jadi setiap tim inovasi sukses yang saya temui menjadikan bagian ini, pemahaman lapangan, sebagai fokus utama mereka.

2. ***Definisikan:*** Setelah langkah pertama yang penting yaitu mendengarkan dan memahami, masalah yang akan dipecahkan perlu disempurnakan dan diubah menjadi sesuatu yang benar-benar dapat dikerjakan oleh orang-orang. Ambil wawasan dari pemahaman Anda dan buat persyaratan yang konkret. Triknya di sini adalah mendefinisikan masalah dengan ketat tetapi hindari memulai pemecahan masalah dengan menyarankan pendekatan.

   Pikirkan setidaknya lima kategori persyaratan ini:

   a. Apa masalah saat ini – apa yang perlu diubah
   b. Apa yang perlu dihasilkan solusinya – apa hasilnya
   c. Apa yang perlu dikonsumsi solusinya – apa saja masukannya
   d. Di mana solusinya perlu dioperasikan – di mana solusinya digunakan
   e. Untuk siapa solusinya perlu digunakan – siapa yang akan menggunakannya

   Dengan memperluas contoh laporan harian kita, kita dapat mendefinisikan masalah sebagai:

   a. Perubahan: gangguan dan kurangnya pemahaman
   b. Hasil: laporan harian yang konsisten dan lengkap
   c. Masukan: informasi dari pengawas dan mandor
   d. Konteks penggunaan: API perangkat lunak CM
   e. Pengguna: manajer proyek, manajer konstruksi, keuangan

   Jadi, definisi masalahnya adalah kita perlu menemukan cara untuk mendapatkan informasi dari pengawas dan mandor kepada manajer proyek mereka, melalui API perangkat lunak manajemen konstruksi mereka, dengan cara yang meminimalkan gangguan mereka, dan memperjelas pentingnya informasi tersebut.

   Proses ini juga harus menghasilkan kriteria. Langkah-langkah yang disebutkan akan membantu untuk melakukan ini, dan Anda harus setuju sebagai sebuah kelompok bahwa kriteria Anda sudah memadai, bahwa Anda telah memenuhi semua persyaratan yang diperlukan tetapi tidak lebih. Bagian terakhir itu penting – benar-benar berdebat apakah ada persyaratan Anda yang tidak perlu, karena Anda akan membutuhkannya nanti. Persyaratan tambahan dapat mendiskualifikasi ide-ide yang sebenarnya bagus, dan menyebabkan Anda perlu merevisi persyaratan Anda nanti dalam proses tersebut.

3. ***Ide:*** Ide adalah pembuatan ide untuk memecahkan masalah. Hal ini sering kali paling baik dilakukan dalam "putaran" berpikir, di mana tim berbagi ide, mendiskusikan kekuatan dan kelemahan, dan mencoba merumuskan solusi.

   Brainstorming telah menjadi metodologi pilihan untuk ide semacam ini selama beberapa dekade, karena mudah dan menghargai satu ide utama – menghasilkan ide bukanlah proses yang sama dengan menilai ide, dan kedua proses tersebut harus dipisahkan.

Apakah Anda menyebutnya brainstorming atau hanya rapat, langkah ide perlu disusun sedemikian rupa sehingga ada waktu untuk menghasilkan banyak ide, kemudian langkah terpisah untuk menyaring ide-ide tersebut dan meninjau ide-ide yang tersisa, dan mengulanginya.

Setelah menjalankan banyak hal ini, berikut beberapa petunjuk yang dapat membantu produktivitas:

- Orang-orang sering kali memiliki 10 menit pertama yang sangat produktif, karena mereka hanya mengeluarkan ide-ide yang telah mereka pikirkan selama beberapa saat. Ini sering kali merupakan ide-ide yang bagus.
- Orang-orang sering kali malas, dan akan berhenti setelah beberapa ide. Memberi mereka sejumlah besar ide yang dibutuhkan memaksa mereka untuk benar-benar menggali dan berpikir – 20 adalah angka yang pernah saya gunakan di masa lalu, karena tidak ada orang yang berjalan-jalan dengan 20 ide tentang apa pun, jadi Anda pasti akan memiliki peserta yang mendorong ke area pemikiran baru.
- Ingatlah bahwa kreativitas adalah tentang menghubungkan ide-ide dengan cara-cara baru. Jadi, berikan kesempatan kepada peserta untuk berbagi ide. Di sinilah catatan tempel sering digunakan. Seorang konsultan kreativitas yang saya kenal, Bryan Mattimore, suka meminta peserta berjalan dari satu area yang penuh dengan ide-ide catatan tempel ke area lain, sehingga mereka dapat menyatukan ide-ide.
- Akhiri proses dengan beberapa bentuk pemungutan suara untuk ide mana yang harus menang. Pastikan Anda menyertakan mekanisme untuk memasukkan kriteria dalam proses pemungutan suara. Orang cenderung melupakan beberapa kriteria, dan Anda akhirnya harus kembali dan menyaring ulang jika Anda tidak menjaga kriteria di depan dan di tengah selama proses pemilahan.

Tujuan dari proses ide adalah untuk memiliki sebanyak mungkin finalis yang dapat Anda buat prototipenya – Anda ingin sebanyak mungkin opsi nyata di tangan Anda, tergantung pada jenis solusi yang ingin Anda kembangkan.

4. ***Prototipe***: Kontribusi khas pemikiran desain terhadap proses kreatif dan inovasi secara umum harus berupa fase pembuatan prototipe. Ide-ide yang telah dipilih telah menjalani semua langkah kreatif "tradisional", tetapi sekarang saatnya untuk menerobos lapisan-lapisan tebakan tentang seperti apa ide itu nantinya dan mewujudkannya secara nyata.

   Jelas, jarang sekali mungkin untuk benar-benar membangun sesuatu, baik itu proses atau perangkat lunak baru, atau apa pun. Ada banyak, banyak sekali opsi pembuatan prototipe yang dapat mempersempit kesenjangan antara kenyataan dan pernyataan ide Anda dari langkah #3. Ini bisa berupa pembuatan bagan organisasi, melakukan permainan peran, atau menulis naskah tentang bagaimana sesuatu dapat bekerja. Alasan mengapa pembuatan prototipe sangat ampuh adalah karena ide hanyalah abstraksi, yaitu figur-figur yang telah dihilangkan atau diabaikan semua bentuk dan

sudut tajamnya. Kehidupan nyata penuh dengan sudut tajam dan konsekuensi yang tak terduga. Saat Anda membuat prototipe, Anda harus membuat keputusan tentang menerjemahkan abstrak ke dalam bentuk konkret, dan itu mengajarkan Anda banyak hal tentang ide Anda. Sering kali tindakan pembuatan prototipe mengubah ide, membuatnya lebih baik, membuatnya lebih mungkin berhasil.

Terkadang, hanya membuat prototipe sebuah ide dapat menghancurkannya. Tidak sulit membayangkan memikirkan ulang sebuah proses, lalu menjalankannya dan menyadari bahwa tidak seorang pun akan benar-benar melakukan proses tersebut, atau bahwa langkah tertentu terlalu sulit.

5. ***Uji***: Pembuatan prototipe adalah langkah internal. Setelah Anda membuat prototipe dan cukup bereksperimen untuk membuat ide tersebut terwakili dengan baik, mintalah orang lain untuk mencobanya. Anda melakukan ini karena mereka akan menemukan hal-hal yang tidak Anda pikirkan, baik maupun buruk. Terkadang Anda akan berakhir di sini, tetapi paling sering Anda akan mempelajari hal-hal baru yang akan membuat Anda mengubah pernyataan masalah, atau kriteria baru, dan menambahkan ide-ide baru. Jika demikian, saatnya untuk melakukannya lagi.

   Peningkatan proses akan menjadi aktivitas dengan nilai tambah tertinggi yang Anda lakukan dalam 10 tahun ke depan, jadi memahami cara menghasilkan proses baru sangatlah penting. Setiap teknologi yang Anda adopsi akan memerlukan proses baru untuk mendukungnya, dan perubahan dalam proses yang ada untuk memanfaatkannya secara maksimal. Mudah-mudahan, Anda juga akan memiliki proses untuk menangkap cara-cara tak terduga dalam menggunakan teknologi tersebut.

Mempelajari nilai berpikir melampaui batas, secara lateral, mendobrak asumsi dan berpikir dari prinsip-prinsip awal, dengan sendirinya merupakan keunggulan kompetitif. Melatih tim Anda, di setiap level, dalam pemikiran desain merupakan keunggulan kompetitif lainnya, karena hal itu menanamkan skeptisisme terhadap asumsi dan proses nyata untuk melakukan sesuatu tentangnya.

9.6 INOVASI SEBAGAI PRAKTIK PERUSAHAAN

Apa yang membuat sebuah perusahaan menjadi inovatif? Mari kita kembali ke definisi inovasi kita: "Inovasi adalah kreativitas yang diterapkan pada sebuah produk atau layanan untuk menciptakan nilai yang lebih besar." Dalam konteks perusahaan, itu berarti berpikir kreatif tentang bagaimana perusahaan melakukan apa yang dilakukannya, terus-menerus menciptakan nilai yang lebih besar. Perhatikan bahwa definisi tersebut tidak mengatakan "nilai yang lebih besar bagi pelanggan." Ini karena banyak inovasi perusahaan mengurangi biaya, meningkatkan keselamatan, dan memberikan manfaat lain yang mungkin tidak dilihat oleh pelanggan. Sejauh ini, kita telah meninjau kreativitas dan proses untuk menciptakan ide-ide baru dan mengujinya. Bagian-bagian utama yang hilang adalah mengintegrasikan ide-ide baru ke dalam struktur perusahaan, dan membuatnya menambah nilai dalam skala besar. Bagaimana kita melakukannya? Bagaimana kita membuat puluhan atau ratusan pengawas

menggunakan ide-ide baru, proses-proses baru, dan produk-produk baru? Pertama, mari kita lihat apa yang kita ketahui tentang adopsi produk, karena meskipun personel lapangan bekerja untuk perusahaan, semua orang tahu bahwa mereka memiliki banyak keleluasaan tentang alat apa yang mereka gunakan, dan terutama seberapa baik mereka menggunakannya. Kami telah mempelajari bagaimana produk baru diadopsi setidaknya sejak tahun 1950-an, ketika para peneliti di Iowa mengamati proses yang dilalui petani untuk mulai menggunakan benih jagung hibrida yang lebih unggul daripada benih normal. Ada anggapan umum bahwa adopsi apa pun terjadi secara bertahap, dalam suatu proses, yang jelas benar kecuali jika perusahaan mengamanatkannya, dan bahkan ada yang tertinggal. Namun, yang benar-benar penting adalah akan ada kelompok-kelompok yang berbeda - ada sebuah buku yang diterbitkan beberapa dekade lalu yang mengusulkan lima kelompok, dan bahkan menyarankan persentase populasi tertentu yang diwakili oleh setiap kelompok. Tidak ada bukti sama sekali bahwa persentase ini nyata, jadi jangan khawatir tentang hal itu - dan juga tidak ada bukti sama sekali bahwa benar-benar ada lima kelompok pengadopsi. Kelompok-kelompok tersebut dikembangkan dari kemudahan statistik yang tidak penting. Kelompok yang umum dipahami adalah “inovator,” “pengguna awal,” “mayoritas awal,” “mayoritas akhir,” dan “penganut lambat.” Saya sebutkan di sini karena Anda mungkin pernah mendengar beberapa di antaranya.

Menurut pengalaman saya, yang pasti berguna adalah memikirkan tiga kelompok, karena proses adopsi mereka akan berbeda dalam hal tindakan. Ada pengadopsi awal, mayoritas, dan yang lamban.

Pengadopsi awal terbuka terhadap ide-ide baru, seperti mempelajari hal-hal baru dan akan toleran terhadap hal-hal yang tidak berjalan sempurna. Mereka sering kali lebih mahir secara teknis daripada rekan-rekan mereka, dan biasanya merupakan sumber inovasi yang dipimpin pengguna yang saya sebutkan sebelumnya. Pengadopsi awal tidak perlu banyak diyakinkan, dan mereka tidak perlu melihat orang lain menggunakan produk tersebut. Cara terbaik untuk memberi insentif kepada pengadopsi awal adalah dengan menghargai minat dan energi mereka dan meminta bantuan mereka dalam melakukan penyesuaian atau penyempurnaan apa pun yang diperlukan untuk mengadaptasi produk ke perusahaan Anda.

Mayoritas tidak tertarik untuk menjadi inovatif, mereka hanya menginginkan hasil yang lebih baik. Ada perhitungan risiko/imbalan di sini yang mendefinisikan mereka sebagai mayoritas – orang-orang yang tidak peduli dengan teknologi tetapi peduli dengan hasil. Kami telah menemukan beberapa kiat yang membantu meyakinkan mayoritas untuk mengadopsi:

1. Orang cenderung mengadopsi hal-hal yang mereka lihat diadopsi orang lain, karena hal itu menurunkan risiko yang dirasakan. Yang menarik adalah bahwa penelitian telah menunjukkan bahwa yang penting bukanlah jumlah orang lain yang kita lihat mengadopsi sesuatu, melainkan proporsinya. Jika saya berada di sebuah ruangan dengan empat orang lain dan tiga orang menggunakan produk perangkat lunak baru, dampaknya akan lebih besar daripada melihat 10 orang di ruangan yang beranggotakan 30 orang.

Implikasi praktisnya adalah mencoba memperkenalkan ide-ide baru dalam konsentrasi geografis atau fungsional daripada mencoba memperkenalkannya dalam lapisan tipis di seluruh perusahaan. Ketika Anda memiliki semua departemen yang menggunakan produk atau proses baru, Anda akan memiliki testimonial dan "bukti sosial" untuk memperkenalkannya ke kelompok yang berdekatan.

2. Produk tersebut perlu terasa lebih solid ketika digunakan oleh mayoritas. Ada fenomena yang dikenal sebagai "jurang pemisah" antara pengadopsi awal dan mayoritas, dan itu sebagian besar tentang jenis risiko tertentu. Pengadopsi awal tidak keberatan menghabiskan waktu mencari tahu cara menggunakan produk baru, tetapi mayoritas tidak. Sebenarnya cukup jarang orang benar-benar berpikir produk baru tidak akan berfungsi, tetapi mereka sangat sering berpikir itu tidak akan berhasil bagi mereka, atau akan sulit dipahami, membuang-buang waktu mereka dan membuat mereka terlihat bodoh. Jadi, ketika beralih dari pengadopsi awal menjadi mayoritas, pastikan Anda telah menyiapkan pelatihan, dan memperbaiki bug apa pun.

Pengadopsi lambat tidak tertarik pada perubahan, terlepas dari risiko/imbalannya. Mereka bekerja untuk bekerja, dan menyukai kepastian melakukan hal-hal dengan cara yang sama berulang-ulang. Dalam banyak kasus, pekerjaan hanyalah pekerjaan dan mereka tidak tertarik untuk berkembang atau belajar. Kelompok yang lamban akan mengadopsi karena tiga alasan: karena tidak melakukannya membuat mereka merasa konyol, mereka perlu mengadopsi teknologi baru untuk bekerja dengan orang lain, atau manajemen telah memaksa mereka.

Masing-masing dari ketiga kelompok ini dapat memiliki ukuran yang berbeda, dan kegiatan seperti hackathon dan kontes yang dibahas sebelumnya akan membantu menumbuhkan pengadopsi awal dan mengecilkan kelompok yang lamban.

Yang terpenting adalah orang yang bukan pengadopsi awal melihat tiga hal ini:

1. Banyak orang lain yang menggunakan produk baru.
2. Orang-orang yang seperti mereka menggunakannya. Hal ini terutama berlaku untuk usia, karena Generasi Milenial umumnya akan merasa nyaman dengan beberapa teknologi yang tidak dimiliki Generasi Baby Boomer. Generasi Baby Boomer perlu melihat Generasi Baby Boomer menggunakan sesuatu yang baru.
3. Proses adopsi mudah dan didukung dengan baik melalui pelatihan.

Pasar konsumen telah mencoba gagasan "influencer" sebagai cara untuk membuat orang membeli atau mengadopsi produk baru, dan itu tidak pernah terbukti benar-benar berhasil, meskipun mungkin terasa lebih mudah untuk hanya meminta satu orang menjadi perwakilan Anda. Masalah dengan para influencer atau juara adalah mereka hanya mengenal sedikit orang, jadi tidak apa-apa jika mereka membantu, dan jelas para pengadopsi awal harus dibina dan dimanfaatkan untuk peluncuran yang lebih besar, tetapi itu bukan pengganti kampanye yang cermat dan terpadu yang mencakup tiga elemen yang dibahas sebelumnya (lihat banyak orang lain, seperti mereka, yang didukung dengan baik oleh pelatihan). Adopsi produk atau proses baru adalah manajemen perubahan, dan memerlukan kampanye internal, seperti perubahan strategis perusahaan besar lainnya.

Tim Inovasi

Kontraktor umum dan kontraktor khusus yang lebih besar telah menciptakan tim inovasi dalam dekade terakhir, yang didedikasikan untuk transformasi digital, meskipun mereka tidak selalu menyebutnya demikian. Tim-tim ini biasanya melalui tiga tahap:

- *Tahap 1:* CEO menciptakan tim dan memberi mereka sejumlah anggaran. Mereka merekrut orang untuk tim tersebut, dan menghabiskan waktu sekitar satu tahun untuk merancang pendekatan mereka. Tahap ini biasanya melibatkan drone dan beberapa teknologi canggih lainnya yang tidak digunakan.
- *Tahap 2:* Tim kembali berfokus untuk menarik solusi yang menurut mereka dibutuhkan perusahaan, sering kali mencari juara di lapangan. Terkadang ini berhasil, sering kali tidak.
- *Tahap 3:* Tim sekarang telah mengembangkan kegiatan penjangkauan, seperti kontes dan hackathon. Mereka telah menemukan pengadopsi awal, pengawas, atau mandor yang terus-menerus mencari keunggulan, sebagai cara untuk meningkatkan. Mereka menciptakan mekanisme umpan balik dan saran untuk mendapatkan ide dari lapangan, dan cara formal untuk mempelajari perangkat lunak dan teknologi lain yang ada di luar sana. Kunci keberhasilan dalam langkah ini adalah bahwa tim inovasi menanggapi kebutuhan di lapangan, alih-alih mengeluarkan ide yang belum diminta.

Inovasi adalah kunci kesehatan dan pertumbuhan konstruksi. Perusahaan yang merangkul inovasi sebagai kompetensi inti dan serangkaian proses, alih-alih hasil yang diharapkan, akan menjadi pemenang dalam industri konstruksi masa depan.

BAB 10
DESAIN KONSEPTUAL

10.1 APA YANG DIMAKSUD DENGAN DESAIN KONSEPTUAL?

Desain Konseptual adalah tahap awal dalam perancangan sistem atau produk yang berfokus pada identifikasi kebutuhan dan gambaran umum dari solusi tanpa memperhatikan detail teknis implementasinya. Dalam desain konseptual, konsep utama dan struktur dasar ditetapkan untuk memberikan arah bagi pengembangan lebih lanjut. Tujuannya adalah untuk merumuskan ide besar, fungsi utama, dan alur sistem secara menyeluruh sebelum memasuki tahap desain lebih rinci seperti desain logis dan fisikal. Selain itu desain konseptual juga merupakan proses mengumpulkan, menganalisa dan memprioritaskan perspektif dari pemakai dan bisnis dari suatu permasalahan dan solusi terhadap permasalahan tersebut yang kemudian membuat representasi level tinggi suatu solusi.

Hasil Akhir Konseptual Desain

a. **Memahami Masalah Bisnis Untuk Dipecahkan:** Hal ini merujuk pada proses identifikasi, analisis, dan klarifikasi tantangan atau kebutuhan spesifik yang dihadapi oleh sebuah organisasi atau perusahaan. Ini adalah langkah penting yang harus dilakukan sebelum merancang solusi dalam bentuk sistem, produk, atau layanan baru. Tujuannya adalah untuk memastikan bahwa solusi yang dikembangkan akan relevan, efektif, dan memberikan nilai tambah bagi bisnis. Memahami masalah bisnis untuk dipecahkan adalah langkah krusial dalam proses desain konseptual, yang memungkinkan tim untuk merancang solusi yang tepat, efektif, dan dapat diimplementasikan dengan sukses.
b. **Memahami Kebutuhan Bisnis, Kebutuhan Customer dan Pengguna Akhir:** Poin ini merupakan poin penting karena poin ini adalah proses identifikasi dan analisis yang bertujuan untuk memahami berbagai jenis kebutuhan yang berpengaruh pada pengembangan suatu solusi, baik itu sistem, produk, atau layanan. Pemahaman yang mendalam tentang ketiga elemen ini sangat penting untuk memastikan bahwa solusi yang dirancang tidak hanya memenuhi tujuan bisnis tetapi juga relevan dan bermanfaat bagi pengguna akhir. Secara keseluruhan, memahami kebutuhan bisnis, kebutuhan customer, dan pengguna akhir adalah langkah penting dalam proses desain konseptual yang membantu dalam menciptakan solusi yang efektif, relevan, dan bermanfaat.
c. **Menggambarkan Target Status Masa Depan Setelah Solusi Dibuat:** Konteks ini merujuk pada proses menggambarkan visi dan tujuan yang ingin dicapai setelah implementasi solusi yang telah dirancang. Ini melibatkan pemetaan kondisi ideal atau hasil yang diharapkan setelah sistem, produk, atau layanan baru diterapkan. Tujuannya adalah untuk memberikan panduan yang jelas dan terukur mengenai apa yang diharapkan dari solusi tersebut dan bagaimana dampaknya terhadap organisasi,

pengguna, dan proses bisnis secara keseluruhan. Secara keseluruhan, menggambarkan target status masa depan setelah solusi dibuat adalah langkah penting dalam desain konseptual, yang membantu mengarahkan pengembangan dan memastikan bahwa solusi yang diimplementasikan memberikan nilai tambah yang signifikan bagi organisasi dan penggunanya.

10.2 TAHAPAN-TAHAPAN DESAIN KONSEPTUAL

Pada tahap research tugas-tugas yang perlu dilakukan adalah memperoleh jawaban atas pertanyaan inti, mengidentifikasi inti proses bisnis dan aktivitasnya, mempriotaskan aktivitas dan proses serta memvalidasi, menyaring, dan memperluas draft kebutuhan, use case dan usage scenario. Dalam tahap memperoleh jawaban atas pernyataan inti merujuk pada proses analisis dan evaluasi yang bertujuan untuk menjawab pertanyaan-pertanyaan kunci yang terkait dengan kebutuhan, tujuan, dan fungsi dari solusi yang akan dirancang. Pernyataan inti ini sering kali mencakup aspek-aspek fundamental yang menjadi dasar dari proyek atau sistem yang sedang dikembangkan, dan memperoleh jawaban yang jelas membantu memastikan bahwa semua pihak terlibat memahami arah dan tujuan proyek dengan baik.

Pada tahap selanjutnya yaitu mengidentifikasi inti proses bisnis dan aktivitasnya, tahap ini merujuk pada langkah-langkah yang diambil untuk memahami dan mendefinisikan elemen-elemen kunci yang terlibat dalam proses bisnis suatu organisasi. Ini melibatkan analisis mendalam tentang bagaimana bisnis beroperasi, aktivitas apa saja yang dilakukan, dan bagaimana aktivitas tersebut saling terkait. Tujuannya adalah untuk memastikan bahwa solusi yang dirancang dapat mendukung atau meningkatkan proses bisnis yang ada, serta memenuhi kebutuhan pengguna dan organisasi secara keseluruhan. Secara keseluruhan, *mengidentifikasi inti proses bisnis dan aktivitasnya* adalah langkah penting dalam desain konseptual yang membantu membangun fondasi yang kuat untuk pengembangan solusi yang efektif. Ini memastikan bahwa solusi yang dirancang tidak hanya memenuhi kebutuhan pengguna tetapi juga meningkatkan kinerja dan efisiensi proses bisnis yang ada.

Pada tahap mempriotitaskan aktivitas dan proses adalah langkah di mana tim desain atau pengembang menentukan mana aktivitas dan proses yang paling penting untuk difokuskan dan ditangani dalam pengembangan solusi. Proses ini melibatkan analisis dan evaluasi terhadap berbagai aktivitas dalam konteks bisnis untuk memastikan bahwa sumber daya, baik itu waktu, tenaga, maupun biaya, digunakan secara efisien dan efektif untuk mencapai hasil yang diinginkan. Ini membantu memastikan bahwa solusi yang dirancang tidak hanya efektif tetapi juga relevan dengan kebutuhan bisnis dan pengguna, sehingga memberikan hasil yang maksimal.

Tahap terakhir yaitu memvalidasi, menyaring, dan memperluas draft kebutuhan, use case dan usage scenario merupakan proses kritis yang dilakukan untuk memastikan bahwa semua kebutuhan, skenario penggunaan, dan use case yang telah dikembangkan akurat, lengkap, dan relevan untuk sistem atau solusi yang sedang dirancang. Langkah ini bertujuan untuk menyaring informasi yang tidak relevan, memperdalam pemahaman tentang

kebutuhan pengguna, serta memastikan bahwa semua aspek penting telah dipertimbangkan. Proses ini membantu memastikan bahwa solusi yang dirancang tidak hanya memenuhi kebutuhan yang teridentifikasi tetapi juga relevan dan dapat diterapkan dalam konteks nyata, sehingga meningkatkan kemungkinan keberhasilan proyek.

Dalam proses analisis ada beberapa hal yang bisa dilakukan yaitu:

a. **Meninjau Hasil Riset:** Langkah di mana tim desain atau analisis mengevaluasi, menginterpretasikan, dan memanfaatkan data dan informasi yang telah dikumpulkan dari berbagai sumber penelitian. Proses ini penting untuk memahami konteks, kebutuhan pengguna, dan tantangan yang dihadapi oleh sistem atau solusi yang sedang dirancang. M*eninjau hasil riset* adalah langkah penting dalam proses analisis pada tahapan desain konseptual. Proses ini memberikan dasar yang kuat untuk memahami kebutuhan pengguna dan tantangan yang dihadapi, serta memastikan bahwa solusi yang dikembangkan benar-benar efektif dan relevan dengan konteks yang ada.
b. **Menyaring Kandidat Requirement:** berarti proses menyeleksi persyaratan (requirement) atau kebutuhan yang sudah dikumpulkan untuk memastikan hanya yang relevan, tepat, dan realistis yang digunakan dalam perancangan solusi. Pada tahap ini, tim desain atau perancang konsep biasanya mengevaluasi berbagai kandidat requirement yang mungkin sebelumnya dikumpulkan dari pemangku kepentingan, riset pasar, atau analisis kebutuhan fungsional.
c. **Mendokumentasikan dan Memodelkan Konteks, Workflow, Task Sequence dan Hubungan Dengan Lingkungan:** Dalam tahapan desain konseptual tahapan ini adalah proses dimana tim perancang menganalisis dan menggambarkan konteks operasional dari sistem atau produk yang akan dikembangkan. Tahap ini bertujuan untuk memahami dan menyusun secara sistematis bagaimana pengguna berinteraksi dengan sistem dalam skenario yang realistis, serta bagaimana sistem berhubungan dengan elemen lain di lingkungannya.

Adapun beberapa hal yang bisa kita lakukan dalam proses optimasi yakni:

a. **Mengoptimalkan Konsep Solusi:** Dalam konteks tahapan desain konseptual, *tahapan ini* merujuk pada proses memperbaiki dan menyempurnakan konsep awal dari solusi yang telah dikembangkan, agar mencapai efektivitas dan efisiensi maksimal. Proses ini melibatkan evaluasi, analisis, dan penyesuaian terhadap berbagai aspek desain untuk memastikan bahwa solusi tidak hanya memenuhi kebutuhan dasar, tetapi juga optimal dalam hal performa, biaya, daya tahan, dan faktor-faktor lain yang relevan.
b. **Memvalidasi dan Mengetes Proses Bisnis Yang Telah Dikembangkan:** Ini adalah tahap dimana tim desain menguji dan memastikan bahwa proses bisnis yang dirancang benar-benar berfungsi sesuai dengan tujuan yang diinginkan. Ini berarti memastikan bahwa seluruh alur kerja, tugas, dan interaksi dalam proses bisnis berjalan efektif, efisien serta sesuai dengan kebutuhan bisnis dan pengguna. Dengan melakukan validasi dan pengujian proses bisnis yang telah dikembangkan, desainer dan tim pengembang memastikan bahwa solusi yang dirancang dapat berjalan lancar, efektif,

dan sesuai dengan kebutuhan bisnis serta pengguna, sehingga lebih siap untuk implementasi.

Fase Planning

Setiap tahapan ini merupakan bagian dari proses perencanaan dan pengembangan proyek yang bertujuan untuk memastikan semua elemen terpenuhi sebelum peluncuran atau penerapan solusi. Setiap fase ini membutuhkan persetujuan dan validasi dari setiap pihak yang terlibat agar memastikan setiap tahap berjalan sesuai rencana dan mengurangi risiko kesalahan di fase akhir.

Menuju Fase Planning

Ada dua fase utama dalam perencanaan proyek, yaitu *Envisioning Phase* dan *Planning Phase*, beserta elemen-elemen penting yang diperlukan di masing-masing fase.

1. **Envisioning Phase (Fase Pencitraan)**

   Fase ini bertujuan untuk mengembangkan gambaran awal mengenai proyek, mengidentifikasi kebutuhan dasar, dan menyusun konsep solusi. Berikut adalah tahapan yang dilakukan pada fase ini:

   - **High-level Requirements** (Kebutuhan Tingkat Tinggi): Di tahap ini, kebutuhan utama dari proyek diidentifikasi. Kebutuhan tingkat tinggi menggambarkan gambaran besar dari tujuan proyek, apa yang diharapkan dari solusi yang dikembangkan, dan hasil apa yang ingin dicapai.
   - **High-level Scenarios** (Skenario Tingkat Tinggi): Setelah kebutuhan utama ditetapkan, skenario tingkat tinggi dikembangkan. Ini adalah deskripsi singkat tentang bagaimana solusi akan digunakan dalam situasi yang khas. Skenario ini membantu tim untuk lebih memahami aplikasi praktis dari solusi yang sedang direncanakan.
   - **Solution Concept** (Konsep Solusi): Pada tahap ini, konsep solusi mulai dirumuskan. Konsep ini mencakup pendekatan atau metode yang akan digunakan untuk mencapai tujuan proyek. Di sini, beberapa pendekatan mungkin dipertimbangkan (misalnya, Pendekatan A, Pendekatan B, dan Pendekatan C) untuk menentukan pendekatan mana yang paling sesuai dengan kebutuhan proyek.

2. **Planning Phase (Fase Perencanaan)**

   Setelah fase pencitraan selesai, proyek masuk ke fase perencanaan, di mana perincian lebih lanjut dilakukan untuk memastikan solusi dapat diterapkan secara efektif. Tahapan-tahapan di fase ini adalah sebagai berikut:

   - **Detailed Requirements** (Kebutuhan Rinci): Kebutuhan proyek diperinci lebih lanjut berdasarkan kebutuhan tingkat tinggi. Di sini, semua kebutuhan teknis, fungsional, dan operasional dijabarkan secara detail agar setiap anggota tim memahami apa yang diperlukan untuk membangun solusi yang sesuai.
   - **Detailed Usage Scenarios** (Skenario Penggunaan Rinci): Setelah kebutuhan diperinci, skenario penggunaan yang lebih mendetail disusun. Skenario ini menggambarkan bagaimana pengguna akan berinteraksi dengan solusi secara

mendalam. Skenario rinci ini juga membantu mengidentifikasi potensi masalah dan kebutuhan tambahan yang mungkin tidak terlihat pada tahap sebelumnya.

- **Functional Specification** (Spesifikasi Fungsional): Di tahap ini, spesifikasi fungsional dari solusi dibuat. Spesifikasi ini mendefinisikan fitur dan fungsi yang harus dimiliki oleh solusi untuk memenuhi kebutuhan yang telah diidentifikasi. Spesifikasi fungsional kemudian digunakan sebagai panduan selama proses pengembangan.
  Pada akhir tahap ini, tim harus menyusun *Design Documents* yang mencakup:
  - ✓ **Conceptual** (Konseptual): Desain konseptual yang menjelaskan ide-ide dasar dari solusi.
  - ✓ **Logical** (Logis): Desain logis yang menggambarkan struktur hubungan antar komponen solusi.
  - ✓ **Physical** (Fisik): Desain fisik yang menjelaskan elemen teknis dan spesifik, seperti pemilihan perangkat keras dan perangkat lunak yang diperlukan.

Dua fase ini membentuk fondasi dari keseluruhan perencanaan proyek. Envisioning Phase bertujuan untuk memahami kebutuhan dan konsep dasar dari proyek, sementara Planning Phase lebih berfokus pada perincian dan penetapan spesifikasi agar solusi yang dikembangkan sesuai dengan kebutuhan dan tujuan proyek. Setiap elemen dalam fase ini saling mendukung agar proyek dapat berjalan dengan terarah dan efektif.

Desain Proses

Proses desain tahap planning terdiri dari conceptual desain, logical desain dan pyschal desain.

1. Conceptual Desain:

Desain konseptual adalah proses di mana konsep dasar dari sistem atau solusi dibentuk dan diuraikan. Fokusnya adalah menciptakan gambaran besar dari sistem atau proyek yang akan dikembangkan. Dalam tahapan ini, ide-ide awal dikumpulkan, dan sketsa kasar atau diagram yang menggambarkan fungsi utama dari solusi dirancang. Desain konseptual membantu semua pemangku kepentingan memahami tujuan proyek dan bagaimana solusi akan memenuhi kebutuhan yang ada. Adapun beberapa tujuan konseptual desain yaitu:

- Mengidentifikasi kebutuhan pengguna dan tujuan proyek.
- Menyusun struktur dasar dari sistem atau solusi.
- Menggambarkan bagaimana setiap bagian dari sistem akan berinteraksi untuk mencapai tujuan.
- Memberikan panduan dan arah yang jelas sebelum melangkah ke desain teknis yang lebih detail.
- Mengkomunikasikan ide-ide utama kepada tim pengembang dan pemangku kepentingan, sehingga semua pihak memiliki pemahaman yang sama.

Ada beberapa langkah yang bisa kita lakukan untuk memastikan desain sesuai dengan kebutuhan proyek. Langkah pertama adalah memahami kebutuhan pengguna dan tujuan proyek. Ini melibatkan pengumpulan informasi mengenai masalah yang akan diselesaikan oleh

solusi, harapan dari pengguna, dan spesifikasi umum yang diinginkan. Setelah kebutuhan pengguna diidentifikasi, analisis terhadap alur kerja atau proses yang terkait dilakukan. Tim harus memahami bagaimana pengguna akan menggunakan sistem, apa yang akan mereka lakukan, dan apa hasil yang diharapkan. Langkah berikutnya adalah membuat model konsep dari solusi.

Ini bisa berupa diagram atau sketsa kasar yang menggambarkan komponen-komponen utama dari sistem serta interaksinya. Model konsep ini harus menunjukkan aliran data, proses utama, dan hubungan antar bagian sistem. Komponen utama dan fungsionalitas inti yang diperlukan untuk memenuhi kebutuhan pengguna dirinci. Pada tahap ini, tim mulai menentukan bagian-bagian utama dari sistem (misalnya, antarmuka pengguna, database, komponen backend) dan fungsi utama dari setiap komponen tersebut. Tim juga perlu mendiskusikan alternatif solusi atau pendekatan yang berbeda untuk mencapai tujuan proyek. Setiap alternatif dinilai berdasarkan kemampuan untuk memenuhi kebutuhan, kelayakan teknis, dan efisiensi.

Setelah model konsep disepakati, semua temuan dan desain konseptual didokumentasikan. Ini termasuk diagram konsep, deskripsi fungsi utama, dan penjelasan mengenai interaksi antar komponen. Dokumentasi ini nantinya akan menjadi acuan bagi desain logis dan fisik yang lebih detail. Desain konseptual sering kali mencakup beberapa komponen utama yang dijelaskan secara kasar, antara lain:

- **User Interface (UI)**: Gambaran dasar tentang bagaimana pengguna akan berinteraksi dengan sistem. Ini melibatkan struktur antarmuka dan elemen yang akan ditampilkan kepada pengguna.
- **Proses Utama atau Workflows**: Penjelasan mengenai proses utama atau alur kerja yang akan diikuti dalam sistem, mulai dari awal hingga akhir.
- **Data Flow**: Gambaran tentang aliran data dalam sistem, termasuk data input, output, dan bagaimana data akan diproses dan disimpan.
- **Interaksi antar Komponen**: Deskripsi dasar tentang bagaimana komponen utama dari sistem akan berinteraksi satu sama lain untuk menjalankan fungsi yang diinginkan.

Desain konseptual adalah langkah awal yang sangat penting dalam tahapan perencanaan proyek. Ini membantu tim untuk menyusun gambaran umum dari sistem yang akan dikembangkan, memahami kebutuhan pengguna, dan menentukan struktur dasar serta komponen utama dari solusi. Dengan menyusun desain konseptual yang baik, proyek dapat berjalan lebih terarah dan mengurangi risiko perubahan besar di kemudian hari.

2. Logical Desain

Logical design adalah tahapan desain di mana fokusnya adalah menggambarkan struktur logis dari sistem yang mencakup data, proses, dan interaksi antar komponen. Pada tahap ini, detail teknis tentang cara implementasi sebenarnya belum diperhatikan; melainkan, tahapan ini menggambarkan bagaimana sistem akan bekerja dari sudut pandang konseptual-logis. Desain logis membantu memastikan bahwa semua kebutuhan pengguna dan proses bisnis tercakup dengan baik sebelum melangkah ke desain fisik. Logical desain memiliki beberapa tujuan penting yaitu:

- Memberikan gambaran detail dari komponen dan interaksi antar komponen dalam sistem.
- Memastikan bahwa kebutuhan pengguna dan persyaratan fungsional dipenuhi oleh struktur logis dari sistem.
- Menyusun rancangan untuk pengolahan data, aliran data, dan struktur basis data, yang semuanya diperlukan untuk memenuhi tujuan sistem.
- Mempersiapkan dasar untuk desain fisik dengan membuat model yang dapat diimplementasikan secara teknis nantinya.

Logical design terdiri dari beberapa komponen utama yang saling terkait untuk membentuk struktur sistem yang utuh. Pertama adalah model data. Pada komponen ini, tim merancang representasi logis dari data yang akan digunakan dalam sistem. Model ini mencakup *ERD (Entity Relationship Diagram)*, yang menggambarkan entitas-entitas utama dalam sistem, atribut dari setiap entitas, serta hubungan di antara mereka. Selain itu, aturan-aturan bisnis juga ditentukan untuk mengatur penggunaan dan batasan data dalam sistem, seperti aturan bahwa setiap anggota hanya boleh meminjam maksimal lima buku sekaligus. Komponen berikutnya adalah model proses, yang mendefinisikan alur proses utama dalam sistem dan bagaimana data diproses pada setiap langkah.

Process model ini sering kali disajikan dalam bentuk *Data Flow Diagram (DFD),* yang menunjukkan bagaimana data mengalir melalui proses atau fungsi dalam sistem, dari input hingga output. DFD ini membantu dalam memahami aliran data secara menyeluruh. Selain DFD, alur kerja atau workflows juga disusun untuk menunjukkan urutan langkah-langkah yang harus diikuti oleh pengguna atau sistem untuk menyelesaikan tugas tertentu. Komponen berikutnya dalam logical design adalah Logical Components and Functions atau komponen dan fungsi logis. Di tahap ini, tim mengidentifikasi dan mendefinisikan fungsi-fungsi logis yang diperlukan oleh sistem, seperti antarmuka pengguna logis yang menggambarkan interaksi pengguna dengan sistem tanpa menguraikan teknologi yang akan digunakan.

Fungsi utama dari sistem seperti login, pencarian data, atau pengolahan transaksi juga ditentukan pada tahap ini. Selain itu, Business Rules and Constraintsatau aturan bisnis dan batasan juga diuraikan. Aturan-aturan ini akan mengatur proses dalam sistem, termasuk batasan akses pengguna atau batasan data yang bisa dimasukkan. Proses desain logis biasanya dimulai dengan analisis kebutuhan fungsional, yaitu mengidentifikasi kebutuhan pengguna dan menentukan fitur yang diperlukan. Langkah selanjutnya adalah menyusun model data logis, seperti ERD, yang menggambarkan struktur data dalam sistem. Setelah itu, tim membuat *Data Flow Diagram (DFD)* untuk memetakan aliran data, yang kemudian dilanjutkan dengan pengembangan diagram alur kerja atau workflow untuk menggambarkan urutan langkah dalam setiap proses utama. Setelah model data dan proses selesai, komponen logis dan fungsi utama sistem ditentukan untuk memastikan sistem dapat memenuhi kebutuhan pengguna.

Langkah terakhir adalah validasi dan tinjauan dengan pemangku kepentingan untuk memastikan bahwa desain logis mencakup semua kebutuhan dan disetujui sebelum melanjutkan ke tahap desain fisik. Sebagai contoh, jika sistem yang dirancang adalah sistem

peminjaman buku di perpustakaan, desain logis mungkin akan mencakup beberapa komponen seperti data model yang terdiri dari entitas “Anggota”, “Buku”, dan “Peminjaman” yang dihubungkan dengan aturan bisnis tertentu, seperti batasan jumlah buku yang bisa dipinjam oleh setiap anggota. Pada sistem process model mungkin mencakup DFD yang menggambarkan proses peminjaman dari input hingga menghasilkan konfirmasi peminjaman. Komponen logis juga mungkin mencakup fungsi-fungsi seperti “Registrasi Anggota”, “Pencarian Buku”, dan “Proses Peminjaman”.

Sedangkan aturan bisnisnya dapat mencakup pembatasan jumlah buku yang bisa dipinjam atau durasi peminjaman. Logical design memiliki peran penting dalam proses desain karena memberikan gambaran rinci tentang bagaimana sistem akan bekerja dan memastikan semua kebutuhan fungsional terpenuhi. Tahapan ini memberikan kerangka kerja untuk desain fisik yang lebih teknis dan mendetail, serta memudahkan komunikasi dengan pemangku kepentingan non-teknis karena desain ini dapat menggambarkan bagaimana sistem memenuhi kebutuhan pengguna tanpa detail teknis yang rumit. Logical design juga meminimalkan risiko terjadinya kesalahan besar di tahap selanjutnya dengan membantu identifikasi masalah lebih awal. Kesimpulannya, logical design adalah tahap penting dalam perencanaan proyek karena memungkinkan tim proyek menciptakan model struktural sistem yang mengatur alur data dan proses secara terperinci, meskipun masih dalam bentuk abstrak. Desain ini menjadi fondasi bagi desain fisik yang akan lebih teknis, sehingga memudahkan pengembang untuk memiliki arah yang jelas saat mengimplementasikan sistem dan meminimalkan risiko perubahan besar di kemudian hari.

3. Pyschal Desain

Desain fisik adalah tahap di mana perencanaan detail tentang cara implementasi sistem dilakukan, termasuk pilihan perangkat keras, perangkat lunak, dan teknologi yang akan digunakan. Pada tahap ini, semua aspek teknis yang diperlukan untuk merealisasikan sistem diuraikan secara spesifik dan rinci. Desain fisik memperhatikan berbagai faktor teknis seperti kapasitas penyimpanan, pemrosesan data, kebutuhan jaringan, dan keamanan sistem, untuk memastikan solusi yang dirancang bisa berfungsi dengan optimal di lingkungan produksi yang sebenarnya. Desain fisik memiliki beberapa tujuan penting dalam proses pengembangan sistem:

- Mengonversi desain logis menjadi arsitektur teknis yang dapat diimplementasikan.
- Menentukan komponen fisik yang dibutuhkan, seperti perangkat keras dan perangkat lunak.
- Menyusun detail teknis, termasuk kebutuhan jaringan, pemrosesan data, dan penyimpanan.
- Menentukan langkah-langkah keamanan untuk melindungi sistem dan data.
- Menjamin bahwa sistem dapat berfungsi secara efisien di lingkungan operasi yang ditentukan.

Desain fisik mencakup beberapa komponen utama yang menguraikan aspek teknis dari implementasi sistem. Pertama, terdapat kebutuhan perangkat keras (hardware requirements). Pada komponen ini, tim menentukan perangkat keras spesifik yang diperlukan agar sistem dapat berjalan optimal, seperti server, prosesor, memori, kapasitas penyimpanan, serta perangkat jaringan. Pemilihan perangkat keras ini berdasarkan kebutuhan kinerja sistem

yang sudah dianalisis pada tahap desain logis. Selanjutnya adalah kebutuhan perangkat lunak (software requirements).

Desain fisik menentukan perangkat lunak yang akan digunakan, baik untuk sistem operasi, platform basis data, hingga alat-alat pendukung seperti perangkat lunak pengembangan. Pemilihan perangkat lunak disesuaikan dengan desain logis dan kebutuhan fungsional yang telah ditetapkan. Kemudian, terdapat desain basis data (database design), di mana struktur basis data dibuat secara lebih rinci, termasuk pembuatan tabel, indeks, serta optimasi kinerja untuk memastikan data dapat dikelola secara efisien. Strategi backup dan pemulihan juga disusun agar data tetap aman dan andal.

Komponen lainnya adalah desain jaringan (network design) yang mencakup perencanaan infrastruktur jaringan seperti router, switch, topologi jaringan, dan kebutuhan bandwidth. Komponen ini bertujuan untuk memastikan data dapat mengalir dengan lancar antar bagian sistem. Selain itu, desain antarmuka pengguna (user interface design) juga dipertimbangkan, terutama dalam aspek teknis seperti pemilihan framework atau bahasa pemrograman untuk membangun antarmuka. Adapun spesifikasi keamanan (security specifications) yang meliputi langkah-langkah perlindungan sistem seperti enkripsi, autentikasi pengguna, pengaturan firewall, dan rencana pemulihan bencana untuk mengantisipasi kerusakan atau akses tidak sah pada data.

Proses desain fisik dimulai dengan menentukan spesifikasi teknis berdasarkan desain logis, termasuk kebutuhan perangkat keras, perangkat lunak, dan jaringan. Setelah itu, tim menyusun rencana infrastruktur teknologi seperti server dan jaringan yang sesuai dengan kebutuhan skalabilitas dan kinerja sistem. Kemudian, struktur basis data dibangun berdasarkan model data logis, namun dengan rincian fisik seperti struktur tabel dan indeks untuk pengelolaan data yang lebih efisien. Selanjutnya, antarmuka pengguna dirancang dengan pemilihan teknologi yang sesuai untuk mengimplementasikan desain antarmuka yang telah direncanakan. Keamanan sistem juga dipastikan melalui langkah-langkah seperti enkripsi, kontrol akses, dan rencana pemulihan.

Langkah terakhir adalah validasi desain fisik, yang meliputi pengujian awal untuk memastikan setiap komponen teknis dapat berfungsi sesuai kebutuhan ketika diimplementasikan. Sebagai contoh, jika sistem yang dirancang adalah sistem peminjaman buku di perpustakaan, desain fisiknya mungkin mencakup pemilihan server dengan kapasitas yang cukup untuk menyimpan data anggota, buku, dan transaksi peminjaman. Sistem operasi yang digunakan pada server, misalnya, dapat berupa Linux, sedangkan basis datanya mungkin menggunakan MySQL atau PostgreSQL. Desain jaringan mungkin meliputi pemilihan topologi yang memungkinkan akses cepat antar perangkat di perpustakaan.

Selain itu, keamanan sistem diimplementasikan dengan autentikasi pengguna dan enkripsi data untuk menjaga kerahasiaan data anggota dan transaksi peminjaman. Pentingnya *Physical Design* terletak pada kemampuannya untuk mengonversi desain logis ke dalam spesifikasi teknis yang siap diimplementasikan. Tahap ini memastikan semua komponen teknis yang dibutuhkan telah tersedia dan dirancang dengan tepat, sehingga sistem dapat berfungsi sesuai spesifikasi yang diinginkan. Physical design juga membantu meminimalkan risiko

kegagalan pada saat implementasi dengan memastikan semua detail teknis sudah direncanakan dan diuji terlebih dahulu. Sebagai kesimpulan, physical design merupakan tahap akhir dalam proses desain yang mempersiapkan sistem untuk implementasi.

Desain ini memberikan spesifikasi detail terkait perangkat keras, perangkat lunak, jaringan, dan keamanan, sehingga sistem dapat diimplementasikan secara efektif dan efisien. Desain fisik yang baik memungkinkan tim proyek untuk memastikan sistem berfungsi optimal dalam lingkungan nyata serta siap digunakan oleh pengguna. Dalam desain proses, masing masing proses bergantung pada dengan proses-proses lainnya. Artinya, setiap proses dalam desain sistem atau alur kerja saling terkait dan memengaruhi hasil dari proses lain. Pendekatan ini sangat relevan dalam manajemen proyek, pengembangan sistem, atau proses bisnis, di mana setiap tahapan saling berhubungan dan bergantung satu sama lain untuk mencapai tujuan akhir dengan sukses.

Setiap proses tidak berjalan secara paralel atau bersamaan namun simultan yang berarti berarti bahwa meskipun berbagai proses saling bergantung dan bekerja dalam alur yang sama, mereka tidak harus berlangsung secara fisik pada saat yang sama (paralel). Sebaliknya, proses-proses tersebut dapat terjadi secara *simultan*, yang berarti setiap proses terjadi secara berurutan atau mengikuti pola tertentu namun dalam waktu yang berdekatan, seolah-olah terjadi hampir bersamaan, dengan saling memengaruhi hasil satu sama lain. Karena setiap proses memiliki titik awal dan titik akhir yang berbeda.

Contoh Draft Statement Requirment Pada Fase Envisioning

Dalam tahapan desain konseptual, fase Envisioning adalah langkah awal yang bertujuan untuk memahami kebutuhan utama dari proyek atau produk yang sedang dikembangkan. Pada tahap ini, tim perancang atau analis bisnis berusaha mendapatkan gambaran yang jelas mengenai apa yang dibutuhkan oleh pengguna atau pemangku kepentingan. Berikut adalah daftar contoh requirement yang teridentifikasi dalam gambar:

1. **Identifikasi Pelanggan Berdasarkan Produk Dan Lokasi:** Requirement pertama ini bertujuan untuk mengklasifikasikan pelanggan berdasarkan produk yang mereka beli serta lokasi mereka. Dengan identifikasi ini, perusahaan bisa mengetahui produk yang diminati di berbagai lokasi dan memahami perilaku konsumen sesuai wilayahnya.
2. **Identifikasi Penurunan Penjualan Pada Pelanggan:** Requirement kedua berfokus pada mendeteksi adanya penurunan penjualan di kalangan pelanggan tertentu. Identifikasi ini penting untuk memahami penyebab turunnya penjualan dan mengembangkan strategi untuk mempertahankan pelanggan atau meningkatkan daya tarik produk.
3. **Identifikasi Pelanggan Potensial:** Requirement ketiga adalah mengidentifikasi pelanggan potensial yang memiliki kemungkinan tinggi untuk membeli produk di masa depan. Dengan informasi ini, perusahaan bisa lebih fokus dalam mengarahkan strategi pemasaran dan penjualan kepada pelanggan yang berpotensi memberikan keuntungan lebih besar.
4. **Identifikasi Pembeli Terbanyak:** Requirement keempat menekankan pada pengenalan pelanggan atau kelompok pelanggan yang memiliki frekuensi pembelian tertinggi. Hal ini memungkinkan perusahaan untuk memahami profil pembeli utama mereka dan

merencanakan strategi untuk menjaga atau meningkatkan loyalitas pelanggan tersebut.

Keempat requirement ini adalah contoh kebutuhan informasi yang perlu didokumentasikan di fase Envisioning. Pendokumentasian requirement semacam ini membantu dalam merancang sistem atau produk yang selaras dengan kebutuhan bisnis dan mendukung keputusan strategis di masa depan. Dengan memiliki pemahaman yang jelas tentang requirement tersebut, tim proyek dapat lebih mudah mengembangkan solusi yang efektif dan tepat sasaran.

Contoh Restate Requirment Pada Fase Planning

Pada fase ini, requirement yang sudah diidentifikasi dalam tahap Envisioning dipecah menjadi sub-requirement yang lebih terperinci. Proses ini bertujuan untuk mendapatkan pemahaman yang lebih mendalam mengenai kebutuhan bisnis dan memastikan bahwa setiap requirement memiliki cakupan yang jelas dan spesifik. Dengan restate requirement, tim dapat memastikan bahwa setiap kebutuhan memiliki spesifikasi rinci yang mudah diimplementasikan pada tahap berikutnya. Berikut adalah rincian dari requirement yang tercantum:

- **Dapat Menganalisa Data Pelanggan:** Requirement utama ini mengacu pada kemampuan untuk melakukan analisis data pelanggan secara keseluruhan. Hal ini mencakup sub-requirement yang lebih spesifik untuk memberikan analisis yang mendalam.
- **Dapat Menganalisa Keuntungan Berdasarkan Produk**: Ini menitikberatkan pada analisis keuntungan yang dihasilkan dari masing-masing produk. Dengan informasi ini, perusahaan dapat mengidentifikasi produk yang paling menguntungkan.
- **Dapat Menganalisa Keuntungan Berdasarkan Pelanggan:** Requirement ini berfokus pada kemampuan untuk menganalisa keuntungan berdasarkan pelanggan tertentu, sehingga perusahaan dapat mengidentifikasi pelanggan yang berkontribusi besar terhadap pendapatan.
- **Dapat Menganalisa Keuntungan Berdasarkan Daerah:** Analisis keuntungan dipecah berdasarkan lokasi atau daerah pelanggan. Hal ini memungkinkan perusahaan memahami kontribusi keuntungan dari berbagai daerah untuk membantu merancang strategi regional.
- **Dapat Mengurutkan Data Pelanggan (Asc/Des):** Requirement ini bertujuan agar sistem dapat mengurutkan data pelanggan, baik secara ascending (menaik) maupun descending (menurun). Pengurutan ini berguna untuk analisis dan pemahaman lebih baik mengenai prioritas pelanggan.
- **Dapat Mengidentifikasi Tren Penjualan:** Requirement ini bertujuan untuk memungkinkan sistem mengidentifikasi pola atau tren dalam penjualan, sehingga perusahaan dapat membuat proyeksi dan strategi pemasaran yang lebih efektif. Dapat Mengidentifikasi Penurunan Penjualan, hal ini spesifik pada identifikasi penurunan penjualan. Dengan demikian, perusahaan dapat dengan cepat mengetahui adanya tren negatif dan mengambil tindakan untuk memperbaiki situasi tersebut.

Dengan restate requirement ini, tim pengembang dan pemangku kepentingan memiliki gambaran yang lebih terperinci mengenai kebutuhan fungsional dari sistem yang akan dibangun. Proses ini memastikan bahwa semua kebutuhan dipahami secara spesifik dan mendalam, yang akan mempermudah tahap implementasi dan pengembangan selanjutnya. Selanjutnya, yang perlu dilakukan adalah melakukan pengelompokan kebutuhan-kebutuhan tersebut menjadi empat kategori yaitu:

1. Kategori kebutuhan pemakai yang mencakup kebutuhan pemakai nonfungsional seperti bagaimana interaksi user dengan solusi. Yang termasuk dalam kebutuhan pemakai adalah antarmuka atau tampilan aplikasi, performa aplikasi dan training pemakai. Contoh kebutuhan pemakai diantaranya: kasir harus bisa menangani lebih dari satu transaksi permenit dan pelanggan harus menyelesaikan pembelian sebuah produk melalui website dalam waktu lima menit.
2. Kategori kebutuhan sistem yang mendefinisikan transaksi yang terjadi serta urutannya didalam sistem. Contoh: aplikasi bisnis perusahaan mendukung notifikasi kepada pemakai secara realtime dan harus menerapkan komponen notifkasi yang telah disahkan.
3. Kebutuhan bisnis. Kebutuhan bisnis disini menggambarkan apa yang dibutuhkan perusahaan dan harapannya terhadap solusi. Contoh kebutuhan bisnis diantaranya: manager call-center yang harus memiliki keahlian untuk melihat informasi panggilan terakhir, panggilan saat ini dan rata rata jumlah panggilan untuk setiap operator.
4. Kebutuhan operasi menggambarkan pengoperasian secara maksimal sebuah solusi dan meningkatkan layanan dengan mengurangi kemacetan dan risiko.

Penyaringan use case dilakukan untuk memastikan bahwa skenario-skenario yang dirancang benar-benar relevan dan memenuhi kebutuhan sistem. Proses ini bertujuan untuk memperjelas cakupan use case serta memastikan bahwa setiap use case memiliki peran yang mendukung keseluruhan tujuan proyek atau sistem. Aktivitas-aktivitas utama dalam penyaringan use case meliputi:

1. **Membuat Subordinat Use Case**: Langkah ini melibatkan pemecahan use case utama menjadi beberapa subordinate use case. Dengan cara ini, setiap bagian dari use case yang kompleks dapat dipecah menjadi beberapa skenario yang lebih spesifik dan mudah dikelola. Pemecahan ini membantu tim untuk mengelola dan memahami proses secara lebih terperinci.
2. **Membuat Usage Scenario untuk Setiap Subordinat Use Case**: Setelah use case dipecah menjadi bagian-bagian yang lebih kecil, setiap subordinate use case perlu dilengkapi dengan skenario penggunaan (usage scenario). Usage scenario ini berisi langkah-langkah spesifik yang diambil pengguna atau sistem dalam menjalankan fungsi tersebut. Pembuatan skenario ini membantu menggambarkan alur proses dari setiap subordinate use case.
3. **Memvalidasi Setiap Use Case dan Usage Scenario**: Validasi dilakukan dengan membandingkan setiap use case dan skenario penggunaannya terhadap informasi

yang telah dikumpulkan dari aktivitas wawancara, dokumen-dokumen terkait, serta diskusi langsung dengan pengguna. Langkah ini penting untuk memastikan bahwa setiap use case dan skenario yang telah dibuat benar-benar sesuai dengan kebutuhan pengguna dan persyaratan yang telah ditetapkan.

4. **Menyaring Kebutuhan dengan Use Case yang Telah Divalidasi**: Setelah proses validasi, kebutuhan sistem atau proyek diselaraskan dengan use case yang sudah terverifikasi dan skenario penggunaan yang telah dikembangkan. Penyaringan ini memastikan bahwa hanya use case yang valid dan relevan yang akan diimplementasikan dalam sistem, sehingga desain dan pengembangan selanjutnya dapat berjalan lebih fokus dan efisien.

Melalui langkah-langkah ini, tim proyek memastikan bahwa setiap use case yang dipertimbangkan dalam fase Planning benar-benar diperlukan dan sesuai dengan tujuan sistem yang ingin dibangun. Hal ini membantu dalam menciptakan struktur sistem yang solid, relevan, dan efisien, yang akan mempermudah implementasi dan penggunaan di tahap selanjutnya.

10.3 DESAIN LOGIKAL

Setelah menyelesaikan tahap desain konseptual, tim proyek memasuki tahap desain logikal, yang merupakan langkah penting dalam proses pengembangan sistem. Pada tahap ini, setiap permasalahan dan kebutuhan yang telah diidentifikasi pada tahap sebelumnya dipecah menjadi unit-unit yang lebih kecil, atau yang sering disebut sebagai modul. Pemecahan ini bertujuan untuk memudahkan tim dalam menganalisis, merancang, dan mengelola setiap komponen sistem secara lebih mendetail. Dengan menguraikan permasalahan menjadi modul, tim dapat merumuskan solusi yang lebih tepat sasaran untuk setiap aspek spesifik, memastikan bahwa setiap bagian dari sistem bekerja secara optimal dan terintegrasi dengan baik dalam keseluruhan struktur.

Dalam tahap desain logikal, modul berperan sebagai unit logikal yang memfasilitasi proses abstraksi dari use case dan skenario yang telah dirancang sebelumnya pada tahap desain konseptual. Modul-modul ini bertujuan untuk mengurai fungsi-fungsi utama sistem menjadi bagian-bagian yang lebih spesifik dan terorganisir. Dengan pembagian ini, setiap modul dapat mewakili satu aspek atau fitur tertentu dari sistem yang lebih besar, sehingga memudahkan tim dalam memahami, mengembangkan, dan mengelola komponen tersebut secara lebih mendetail. Pendekatan modular ini juga memungkinkan tim untuk menguji dan mengoptimalkan setiap bagian secara independen sebelum diintegrasikan ke dalam keseluruhan sistem, memastikan bahwa setiap fungsi dapat berjalan sesuai kebutuhan yang telah ditetapkan pada tahap konseptual.

Setelah modul-modul selesai disusun, tim melanjutkan proses desain logikal dengan mengidentifikasi elemen-elemen penting dalam setiap modul. Pada tahap ini, setiap modul dianalisis lebih mendalam untuk menentukan objek (objects) yang akan merepresentasikan entitas tertentu dalam sistem, layanan (services) yang menggambarkan fungsi atau operasi yang dapat dilakukan oleh objek tersebut, serta atribut (attributes) yang berisi karakteristik

atau data yang melekat pada setiap objek. Selain itu, tim juga merancang hubungan (relationship) antar objek untuk menciptakan koneksi dan aliran data yang memungkinkan interaksi antara objek dalam suatu modul dan antara modul-modul lainnya.

Dengan pendekatan ini, setiap komponen dalam sistem dapat dikembangkan secara terstruktur, memastikan bahwa setiap bagian berperan sesuai dengan perannya dalam sistem, dan bahwa keseluruhan sistem dapat berfungsi secara efektif dan terintegrasi. Proses ini membantu tim untuk membangun sistem yang fleksibel, modular, dan mudah diadaptasi terhadap perubahan kebutuhan. Setelah mengidentifikasi objek, layanan, atribut, dan hubungan dalam setiap modul, langkah penting selanjutnya dalam desain logikal adalah menentukan kandidat teknologi yang akan digunakan untuk mengimplementasikan solusi sistem. Pada tahap ini, tim mengevaluasi berbagai opsi teknologi yang sesuai dengan kebutuhan masing-masing modul, termasuk pemilihan perangkat lunak, framework, platform, dan alat pengembangan yang mendukung efisiensi dan performa sistem.

Proses pemilihan teknologi ini mempertimbangkan beberapa faktor kunci, seperti skalabilitas untuk menangani peningkatan pengguna atau data, kompatibilitas dengan sistem yang sudah ada, tingkat keamanan, dan kemudahan dalam pemeliharaan jangka panjang. Dengan memilih teknologi yang tepat sejak awal, tim dapat memastikan bahwa setiap modul tidak hanya berfungsi secara logis, tetapi juga siap diimplementasikan secara teknis dalam lingkungan yang mendukung tujuan keseluruhan sistem. Pendekatan ini membantu meminimalkan risiko yang mungkin muncul pada tahap implementasi dan memastikan bahwa semua komponen teknologi dapat beroperasi secara terpadu dalam struktur sistem yang telah dirancang.

Tabel 10.1 Peran Tim Pada Desain Logik

Peran Tim	Tugas Utama	Tugas Lainnya
Program management	Memastikan bahwa desain sudah sesuai dengan kebutuhan bisnis dan pemakai	Mengakomodir harapan-harapan pemakai
Product management	Bertanggung jawab untuk deliverables desain logik	Mendefinisikan 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 Experience	Mendifinisikan performa user dan solusi yang direkomendasikan	Mendefinisikan rencana edukasi bagi user

Release management	Mengevaluasi feasibility penerapan solusi	Mendefinisikan infrastruktur dan pemasangan solusi

Pertimbangan Desain Logik

Pada tahap desain logikal, tim harus memperhitungkan berbagai aspek penting untuk memastikan bahwa solusi yang dirancang tidak hanya memenuhi kebutuhan fungsional, tetapi juga siap diimplementasikan secara efektif, efisien, dan berkelanjutan. Pertimbangan desain logikal ini mencakup elemen-elemen seperti struktur modular, yang memungkinkan setiap bagian sistem untuk dikembangkan dan diubah secara independen tanpa mengganggu keseluruhan; skalabilitas, agar sistem dapat tumbuh dan menangani peningkatan jumlah pengguna atau data di masa mendatang; serta performa, untuk memastikan setiap modul dapat beroperasi dengan kecepatan dan stabilitas yang optimal.

Selain itu, keamanan menjadi faktor utama untuk melindungi data dan integritas sistem dari ancaman eksternal dan internal. Tim juga mempertimbangkan kemudahan pemeliharaan, yang mencakup kesederhanaan dalam mengidentifikasi, memodifikasi, atau memperbaiki komponen jika diperlukan, guna menjaga sistem tetap operasional dan responsif terhadap kebutuhan baru. Melalui pertimbangan-pertimbangan ini, tim memastikan bahwa desain logikal tidak hanya terstruktur dengan baik, tetapi juga fleksibel, tahan lama, dan mudah dikelola dalam berbagai kondisi operasional. Pendekatan yang komprehensif ini membantu tim meminimalkan risiko teknis dan operasional di tahap implementasi maupun dalam jangka panjang, menjadikan sistem siap menghadapi berbagai tantangan yang mungkin muncul di masa mendatang. Ada 3 pertimbangan dalam pertimbangan desain logik yaitu pertimbangan yang berhubungan dengan bisnis, pertimbangan arsitektur enterprise, pertimbangan teknologi.

Pertimbangan Yang Berhubungan Dengan Bisnis

a. **Feasibillity:** Feasibility merupakan langkah penting dalam tahap desain logikal yang bertujuan untuk menetapkan apakah teknologi yang akan digunakan sesuai dengan kebutuhan bisnis dan memenuhi persyaratan yang telah ditetapkan. Proses ini melibatkan analisis mendalam terhadap berbagai aspek, termasuk teknis, ekonomi, operasional, dan hukum. Dari segi teknis, tim perlu memastikan bahwa teknologi yang dipilih mampu mendukung spesifikasi dan kinerja yang diinginkan, serta dapat diintegrasikan dengan sistem yang sudah ada. Dalam hal ekonomi, analisis biaya dan manfaat sangat penting untuk menilai apakah investasi dalam teknologi tersebut sebanding dengan keuntungan yang akan diperoleh. Aspek operasional juga harus dipertimbangkan, memastikan bahwa tim memiliki keahlian dan sumber daya yang diperlukan untuk mengimplementasikan dan memelihara teknologi tersebut. Selain itu, aspek hukum dan kepatuhan harus dievaluasi untuk memastikan bahwa solusi yang dirancang memenuhi semua regulasi dan standar yang berlaku. Dengan melakukan analisis feasibility secara komprehensif, tim dapat meminimalkan risiko terkait dengan pemilihan teknologi dan memastikan bahwa keputusan yang diambil mendukung tujuan strategis bisnis, serta memberikan nilai tambah yang signifikan bagi organisasi.

b. **Product Cost:** Product Cost merupakan elemen krusial dalam perencanaan anggaran proyek, yang mencakup seluruh biaya yang terkait dengan produksi sistem dan implementasinya. Biaya ini tidak hanya mencakup gaji developer yang terlibat dalam pengembangan perangkat lunak, tetapi juga biaya operasional seperti server yang diperlukan untuk menjalankan aplikasi, serta lisensi reseller yang mungkin diperlukan untuk distribusi produk. Selain itu, tim harus memperhitungkan biaya upgrade sistem yang mungkin diperlukan di masa depan, untuk memastikan bahwa produk tetap relevan dan dapat memenuhi kebutuhan pengguna yang terus berkembang. Kebutuhan awal untuk hardware dan software juga menjadi pertimbangan penting; ini mencakup perangkat keras yang diperlukan untuk pengembangan dan pengujian, serta perangkat lunak yang mendukung proses tersebut. Dukungan teknis dan infrastruktur yang diperlukan untuk menjaga sistem berjalan dengan baik juga harus diperhitungkan dalam total biaya produk. Hal ini termasuk biaya pemeliharaan, pemantauan, dan perbaikan yang mungkin muncul selama siklus hidup produk. Akhirnya, training bagi pengguna dan tim yang akan mengelola sistem menjadi bagian integral dari biaya produk, memastikan bahwa semua pihak yang terlibat memiliki pemahaman yang cukup untuk memanfaatkan teknologi secara efektif. Dengan memperhitungkan semua elemen biaya ini secara menyeluruh, tim dapat membuat proyeksi biaya yang lebih akurat dan memastikan bahwa proyek tetap dalam anggaran yang telah ditetapkan, serta memberikan nilai maksimal bagi organisasi.
c. **Experience:** Experience adalah faktor penting dalam pengembangan dan implementasi sistem, yang mencakup tenaga ahli yang dimiliki oleh tim untuk memenuhi berbagai kebutuhan, seperti training, konsultasi, dan memastikan tingkat kenyamanan penggunaan sistem. Tenaga ahli ini tidak hanya membawa pengetahuan teknis yang mendalam, tetapi juga pengalaman praktis dalam menghadapi tantangan yang mungkin muncul selama proses pengembangan dan penerapan sistem. Dalam konteks training, keahlian yang dimiliki oleh tim sangat penting untuk memberikan pendidikan yang efektif kepada pengguna akhir. Hal ini mencakup kemampuan untuk menyampaikan informasi dengan cara yang mudah dipahami, serta menyediakan materi pelatihan yang relevan dan aplikatif. Pengalaman tim dalam merancang program pelatihan yang efektif akan meningkatkan tingkat adopsi dan pemahaman pengguna terhadap sistem baru. Selain itu, tenaga ahli juga berperan dalam memberikan konsultasi, membantu organisasi dalam merumuskan strategi implementasi yang tepat dan menyesuaikan solusi dengan kebutuhan spesifik bisnis. Mereka dapat memberikan wawasan berharga tentang praktik terbaik dan membantu menghindari kesalahan yang umum terjadi. Tingkat kenyamanan penggunaan sistem juga sangat bergantung pada pengalaman pengguna yang dipengaruhi oleh pelatihan yang mereka terima dan dukungan yang diberikan oleh tenaga ahli. Dengan memiliki tim yang berpengalaman, organisasi dapat memastikan bahwa pengguna merasa percaya diri dan nyaman saat berinteraksi dengan sistem baru, yang pada gilirannya akan meningkatkan produktivitas dan efektivitas operasional. Melalui pendekatan ini,

tim dapat menciptakan lingkungan yang mendukung bagi pengguna untuk menjelajahi dan memanfaatkan teknologi dengan cara yang optimal.

d. Return Of Investment: *Return of Investment* (ROI) adalah metrik yang sangat penting dalam evaluasi keberhasilan suatu proyek, yang menunjukkan seberapa efektif investasi yang dilakukan akan memberikan pengembalian di masa depan. Dalam konteks ini, penting untuk memastikan bahwa setiap investasi yang dilakukan memiliki keterkaitan yang jelas dengan pengembalian yang diharapkan. Analisis ROI membantu tim untuk mengidentifikasi dan mengevaluasi potensi manfaat finansial dari proyek, termasuk peningkatan efisiensi operasional, penghematan biaya, dan peningkatan pendapatan yang dihasilkan setelah implementasi sistem. Proses ini mencakup perhitungan biaya awal yang diinvestasikan, seperti pengembangan, pelatihan, dan biaya infrastruktur, serta estimasi pendapatan tambahan atau penghematan biaya yang akan diperoleh seiring berjalannya waktu. Dengan memiliki pemahaman yang jelas tentang ROI, tim dapat membuat keputusan yang lebih informasional tentang kelayakan proyek dan menentukan prioritas investasi. Selain itu, analisis ROI juga berfungsi sebagai alat komunikasi yang efektif untuk meyakinkan pemangku kepentingan dan manajemen tentang nilai dan manfaat dari investasi yang dilakukan. Tim harus secara berkala mengevaluasi dan memantau ROI sepanjang siklus hidup proyek untuk memastikan bahwa ekspektasi pengembalian investasi tetap relevan dan dapat dicapai. Jika diperlukan, penyesuaian strategi atau taktik harus dilakukan untuk memaksimalkan potensi pengembalian, sehingga memastikan bahwa investasi tidak hanya memberikan manfaat jangka pendek, tetapi juga berkontribusi pada pertumbuhan dan keberlanjutan organisasi dalam jangka panjang.

e. Maturity: Maturity adalah tahap penting dalam siklus hidup produk yang menunjukkan bahwa produk tersebut telah mencapai tingkat penerimaan yang tinggi di masyarakat dan dapat beroperasi dengan stabil serta efisien. Pada tahap ini, produk diharapkan untuk memenuhi kebutuhan dan harapan pengguna dengan baik, yang mencakup kemudahan penggunaan dan aksesibilitas yang baik. Untuk mencapai tingkat kematangan ini, produk harus telah melalui berbagai tahap pengujian dan iterasi, di mana umpan balik dari pengguna sebelumnya dimanfaatkan untuk melakukan perbaikan dan penyesuaian. Stabilitas sistem menjadi sangat penting; produk yang matang seharusnya minim gangguan, dan dapat berfungsi tanpa masalah teknis yang signifikan. Hal ini tidak hanya meningkatkan kepercayaan pengguna, tetapi juga berkontribusi pada loyalitas pelanggan yang lebih tinggi. Kemudahan penggunaan juga merupakan faktor kunci dalam menentukan kematangan produk. Antarmuka yang intuitif dan pengalaman pengguna yang menyenangkan dapat membuat perbedaan besar dalam adopsi dan penggunaan produk. Produk yang dirancang dengan mempertimbangkan pengalaman pengguna cenderung mendapatkan penerimaan yang lebih baik di pasar. Selain itu, adanya dukungan sumber daya yang memadai, baik dari segi dukungan teknis maupun pelatihan, sangat penting untuk membantu pengguna beradaptasi dengan produk. Dukungan ini dapat mencakup dokumentasi

yang jelas, saluran bantuan yang responsif, dan sumber daya pelatihan yang efektif untuk memastikan pengguna merasa nyaman dan mampu memanfaatkan semua fitur yang ditawarkan oleh produk. Dengan kombinasi dari penerimaan masyarakat, stabilitas, kemudahan penggunaan, dan dukungan sumber daya, sebuah produk dapat dianggap matang dan siap bersaing di pasar. Tahap kematangan ini bukan hanya mencerminkan keberhasilan produk, tetapi juga merupakan indikator penting bagi keberlanjutan dan pertumbuhan di masa depan.

f. **Supportability:** Supportability adalah aspek kritis dalam perencanaan dan pengembangan sistem, yang berkaitan dengan pemilihan teknologi yang tepat dan mampu mendukung pengembangan solusi jangka panjang. Dalam konteks ini, supportability tidak hanya mencakup kemampuan teknologi untuk diintegrasikan dengan sistem yang ada, tetapi juga kemudahan dalam pemeliharaan dan pembaruan di masa depan. Pemilihan teknologi yang tepat harus mempertimbangkan faktor-faktor seperti kompatibilitas dengan platform yang sudah ada, kemampuan untuk beradaptasi dengan perubahan kebutuhan bisnis, serta potensi skalabilitas untuk menghadapi pertumbuhan di masa mendatang. Teknologi yang memiliki dukungan komunitas yang kuat atau ekosistem yang mapan cenderung menawarkan lebih banyak sumber daya, dokumentasi, dan bantuan teknis, yang sangat berharga dalam proses pengembangan dan pemeliharaan. Supportability juga berhubungan dengan kemudahan dalam mengatasi masalah yang mungkin muncul setelah sistem diimplementasikan. Sistem yang dirancang dengan baik dan didukung oleh teknologi yang sesuai akan lebih mudah untuk diperbaiki dan dimodifikasi, sehingga meminimalkan downtime dan meningkatkan produktivitas. Dukungan dalam hal pelatihan dan pengembangan keterampilan bagi tim internal juga merupakan bagian integral dari supportability. Dengan memastikan bahwa tim memiliki pengetahuan yang memadai tentang teknologi yang digunakan, organisasi dapat memastikan bahwa sistem dapat dikelola dengan baik dan dioptimalkan sesuai dengan kebutuhan yang berkembang. Secara keseluruhan, dengan memilih teknologi yang mendukung supportability, organisasi dapat membangun solusi yang tidak hanya efektif dalam jangka pendek, tetapi juga berkelanjutan dan responsif terhadap perubahan dalam lingkungan bisnis yang terus berubah, sehingga meningkatkan nilai dan efisiensi operasional dalam jangka panjang.

10.4 PERTIMBANGAN ARSITEKTUR ENTERPRISE

Aplikasi yang dikembangkan harus sesuai dengan prinsip dan tujuan yang diuraikan oleh arsitektur enterprise, untuk memastikan bahwa semua elemen sistem bekerja secara harmonis dalam kerangka kerja yang telah ditetapkan. Arsitektur enterprise memberikan panduan strategis yang mencakup standar, kebijakan, dan kerangka kerja yang diperlukan untuk mencapai tujuan bisnis yang lebih besar. Keselarasan aplikasi dengan arsitektur enterprise sangat penting, karena hal ini menjamin bahwa aplikasi tidak hanya memenuhi kebutuhan fungsional tetapi juga mendukung visi dan misi jangka panjang organisasi. Ini

mencakup penerapan prinsip-prinsip desain yang telah ditentukan, seperti penggunaan teknologi yang konsisten, integrasi yang mulus dengan sistem lain, dan pemenuhan standar keamanan serta kepatuhan yang diperlukan.

Dengan mengikuti arsitektur enterprise, aplikasi dapat mengurangi risiko terjadinya silo informasi, di mana data dan fungsi tidak terintegrasi dengan baik, yang dapat menghambat efisiensi operasional dan pengambilan keputusan. Selain itu, aplikasi yang selaras dengan arsitektur enterprise memungkinkan organisasi untuk lebih mudah beradaptasi dengan perubahan kebutuhan bisnis dan teknologi, karena semua perubahan dapat dilakukan dengan mempertimbangkan dampaknya terhadap keseluruhan arsitektur. Pendekatan ini juga memfasilitasi kolaborasi antar tim dalam organisasi, karena semua pihak memiliki pemahaman yang sama mengenai kerangka kerja dan tujuan yang ingin dicapai. Dalam konteks ini, pengembangan aplikasi menjadi bagian dari strategi yang lebih besar, yang memungkinkan organisasi untuk merespons dengan cepat terhadap dinamika pasar dan kebutuhan pelanggan, serta memaksimalkan nilai dari investasi teknologi yang dilakukan. Dengan demikian, memastikan kesesuaian aplikasi dengan prinsip dan tujuan arsitektur enterprise menjadi langkah penting dalam menciptakan solusi yang efektif dan berkelanjutan. "Rencana ke depan dari arsitektur perusahaan sangat penting dalam menentukan arah dan strategi teknologi yang akan diambil untuk mendukung tujuan bisnis jangka panjang.

Misalnya, memilih sistem operasi Microsoft Windows Server 2003 sebagai perangkat lunak server merupakan keputusan strategis yang diambil dengan mempertimbangkan berbagai faktor, seperti stabilitas, dukungan teknis, dan kompatibilitas dengan aplikasi yang ada maupun yang akan datang. Dengan menetapkan Windows Server 2003 sebagai platform dasar, organisasi dapat memastikan bahwa aplikasi-aplikasi yang dikembangkan di masa depan dapat berjalan dengan baik dalam lingkungan ini, memanfaatkan fitur-fitur yang ditawarkan oleh sistem operasi tersebut. Selain itu, keputusan ini juga mempertimbangkan keberlanjutan operasional dan kemampuan untuk mengintegrasikan solusi baru dengan sistem yang sudah ada, sehingga mengurangi risiko yang terkait dengan migrasi atau penggantian perangkat lunak di masa mendatang. Rencana ini juga mencakup pemahaman tentang bagaimana teknologi yang dipilih dapat mendukung inisiatif bisnis, seperti efisiensi operasional, peningkatan produktivitas, dan pengalaman pengguna yang lebih baik. Dalam konteks ini, pengembangan aplikasi yang sesuai dengan lingkungan Windows Server 2003 akan memberikan fleksibilitas untuk memanfaatkan solusi perangkat lunak yang telah terbukti efektif, sambil memastikan bahwa tim pengembang memiliki keterampilan dan sumber daya yang memadai untuk bekerja dengan teknologi yang dipilih.

Selanjutnya, organisasi harus terus memantau perkembangan teknologi dan tren industri untuk memastikan bahwa arsitektur perusahaan tetap relevan dan responsif terhadap perubahan. Rencana kedepan harus mencakup evaluasi berkala terhadap sistem dan infrastruktur yang ada, serta rencana untuk melakukan pembaruan atau migrasi ke versi yang lebih baru jika diperlukan, untuk menghindari potensi masalah keamanan dan kinerja di masa depan. Dengan pendekatan yang proaktif ini, organisasi dapat memaksimalkan nilai dari investasinya dan tetap kompetitif di pasar yang terus berubah. "Arsitektur enterprise

menggambarkan rencana-rencana saat ini dan masa depan yang dirancang untuk memastikan bahwa semua elemen teknologi dan proses bisnis dalam organisasi berfungsi secara terpadu dan efisien. Dengan menyusun arsitektur enterprise, organisasi dapat memiliki panduan yang jelas mengenai bagaimana sistem, aplikasi, dan infrastruktur TI akan saling terintegrasi, serta bagaimana mereka akan beradaptasi dengan kebutuhan dan tujuan bisnis yang berubah seiring waktu.

Arsitektur ini mencakup berbagai komponen, seperti struktur data, teknologi yang digunakan, proses bisnis, dan standar keamanan yang harus dipatuhi. Selain itu, arsitektur enterprise juga mencerminkan visi strategis organisasi, mengidentifikasi inisiatif jangka panjang yang perlu diambil untuk mencapai tujuan tersebut. Dengan demikian, arsitektur ini berfungsi sebagai peta jalan yang membantu pemangku kepentingan dalam membuat keputusan yang tepat terkait investasi teknologi dan pengembangan sistem. Dalam konteks rencana masa depan, arsitektur enterprise harus mencakup fleksibilitas untuk mengakomodasi perubahan yang mungkin terjadi di pasar atau dalam strategi bisnis organisasi. Ini bisa meliputi adopsi teknologi baru, pergeseran dalam model bisnis, atau perubahan dalam kebutuhan pelanggan. Dengan memiliki fondasi arsitektur yang kuat, organisasi dapat dengan mudah mengintegrasikan inovasi dan meningkatkan respon terhadap peluang atau tantangan yang muncul. Selanjutnya, arsitektur enterprise juga memainkan peran penting dalam memastikan bahwa semua tim dalam organisasi, baik IT maupun non-IT, memiliki pemahaman yang sama tentang tujuan dan arah teknologi yang akan diambil. Hal ini mendorong kolaborasi antar departemen dan memfasilitasi pengambilan keputusan yang lebih baik dan lebih terinformasi.

Secara keseluruhan, arsitektur enterprise yang dirancang dengan baik tidak hanya memberikan panduan teknis, tetapi juga mendukung pencapaian visi dan misi organisasi secara keseluruhan. Teknologi yang dapat berinteraksi dengan sistem lain di dalam organisasi sangat penting untuk menciptakan ekosistem yang terintegrasi dan efisien. Interoperabilitas antar sistem memungkinkan pertukaran data dan informasi yang lancar, yang pada gilirannya mendukung kolaborasi antar tim dan meningkatkan produktivitas. Beberapa contoh teknologi yang dapat berinteraksi dengan sistem lain termasuk Application Programming Interfaces (APIs), yang memungkinkan sistem yang berbeda untuk saling berkomunikasi dan bertukar data dengan cara yang terstandarisasi. API memungkinkan pengembang untuk mengintegrasikan aplikasi baru dengan aplikasi yang sudah ada, sehingga mempercepat proses pengembangan dan mengurangi biaya. Selain itu, penggunaan platform middleware juga dapat memfasilitasi interaksi antara berbagai sistem yang mungkin memiliki arsitektur yang berbeda. Middleware bertindak sebagai penghubung yang menyederhanakan komunikasi antara aplikasi, memastikan bahwa data dapat ditransfer dengan efisien dan aman. Teknologi berbasis cloud juga memainkan peran penting dalam interaksi sistem. Dengan penyimpanan dan layanan yang dihosting di cloud, organisasi dapat dengan mudah menghubungkan berbagai aplikasi dan sistem yang berbasis cloud, memungkinkan akses data secara real-time dari mana saja. Ini juga mendukung penggunaan solusi *Software as a Service* (SaaS), yang sering kali memiliki kemampuan integrasi yang kuat.

Selain itu, protokol komunikasi standar seperti HTTP, MQTT, dan AMQP dapat digunakan untuk memungkinkan pertukaran data antar sistem dengan cara yang efisien dan aman. Penggunaan standar ini memastikan bahwa semua sistem dalam organisasi dapat beroperasi dengan baik satu sama lain, meskipun mungkin dikembangkan dengan teknologi yang berbeda. Dengan memilih teknologi yang mendukung interoperabilitas dan integrasi, organisasi dapat menciptakan lingkungan yang lebih kohesif, yang tidak hanya meningkatkan efisiensi operasional tetapi juga memungkinkan inovasi berkelanjutan. Hal ini penting untuk memastikan bahwa sistem yang ada dapat beradaptasi dengan perubahan kebutuhan bisnis dan teknologi di masa depan, menjaga relevansi dan daya saing organisasi.

Pertimbangan Teknologi

Pertimbangan teknologi dalam pengembangan sistem sangat penting, dan salah satu aspek yang paling krusial adalah keamanan. Keamanan sistem mencakup berbagai elemen yang dirancang untuk melindungi data dan informasi organisasi dari ancaman yang mungkin terjadi. Dalam konteks ini, beberapa komponen utama yang harus diperhatikan adalah authentication, access control, encryption, dan auditing.

Authentication adalah proses yang memastikan bahwa pengguna atau sistem yang mencoba mengakses sumber daya adalah siapa yang mereka klaim. Metode authentication yang kuat mencakup penggunaan kata sandi yang kompleks, otentikasi dua faktor (2FA), dan biometrik, seperti pemindaian sidik jari atau pengenalan wajah. Implementasi sistem authentication yang kuat sangat penting untuk mencegah akses tidak sah dan memastikan bahwa hanya pengguna yang terverifikasi yang dapat mengakses data sensitif. Proses ini juga harus memperhatikan kebijakan pengelolaan kata sandi yang baik, termasuk perubahan rutin kata sandi dan penghindaran penggunaan kata sandi yang lemah.

Access control berfungsi untuk mengatur siapa yang memiliki hak akses ke sistem dan data tertentu. Pengaturan ini dapat berbasis peran (*Role-Based Access Control*/RBAC), di mana hak akses ditentukan berdasarkan peran pengguna dalam organisasi, atau berbasis atribut (*Attribute-Based Access Control*/ABAC), di mana akses ditentukan oleh kombinasi atribut pengguna, data, dan lingkungan. Dengan sistem access control yang tepat, organisasi dapat mencegah penyalahgunaan data dan mengurangi risiko kebocoran informasi. Kebijakan ini harus terus diperbarui dan dikaji ulang secara berkala untuk mencerminkan perubahan dalam struktur organisasi dan kebutuhan bisnis.

Encryption adalah teknik penting dalam melindungi data, baik saat transit maupun saat disimpan. Dengan mengenkripsi data, informasi sensitif menjadi tidak terbaca tanpa kunci dekripsi yang tepat. Ada dua jenis utama encryption: symmetric encryption, di mana kunci yang sama digunakan untuk mengenkripsi dan mendekripsi data, dan asymmetric encryption, yang menggunakan sepasang kunci publik dan privat. Implementasi encryption yang baik merupakan langkah proaktif untuk menjaga kerahasiaan dan integritas data, serta untuk memenuhi kepatuhan terhadap regulasi dan standar keamanan, seperti GDPR dan HIPAA.

Auditing adalah proses pemantauan dan pencatatan aktivitas sistem, yang memungkinkan organisasi untuk melacak dan menganalisis perilaku pengguna serta mendeteksi potensi pelanggaran keamanan. Dengan sistem auditing yang efektif, organisasi

dapat mengidentifikasi dan merespons ancaman keamanan dengan lebih cepat. Auditing juga berfungsi untuk memastikan kepatuhan terhadap kebijakan keamanan yang telah ditetapkan dan membantu dalam evaluasi efektivitas kontrol keamanan yang ada. Data audit dapat memberikan wawasan berharga untuk perbaikan berkelanjutan dalam kebijakan dan prosedur keamanan.

Selain empat aspek utama ini, pertimbangan keamanan juga mencakup kebijakan keamanan informasi yang menyeluruh, pelatihan dan kesadaran pengguna, serta strategi respons insiden yang terencana. Dengan melibatkan seluruh karyawan dalam praktik keamanan yang baik, organisasi dapat menciptakan budaya keamanan yang lebih kuat. Secara keseluruhan, pertimbangan terhadap keamanan dalam pemilihan dan implementasi teknologi sangat penting untuk menjaga keandalan sistem dan melindungi data organisasi. Dengan memperhatikan aspek-aspek seperti authentication, access control, encryption, dan auditing, serta mengintegrasikan pendekatan proaktif lainnya, organisasi dapat membangun fondasi keamanan yang kokoh yang akan mendukung keberlangsungan dan reputasi bisnis dalam jangka panjang. Keseimbangan antara keamanan dan fungsionalitas adalah kunci untuk mencapai solusi teknologi yang efektif dan berkelanjutan.

Standar interaksi layanan adalah pedoman yang mengatur bagaimana berbagai komponen dalam sistem berkomunikasi satu sama lain, serta memastikan interoperabilitas antar layanan yang berbeda. Salah satu contoh penerapan standar ini adalah pemilihan teknologi .NET, yang dirancang untuk mendukung platform dengan berbagai bahasa pemrograman yang berbeda, seperti C#, VB.NET, dan F#.

Dengan menggunakan .NET, pengembang dapat menciptakan aplikasi yang lebih mudah diintegrasikan dengan sistem lain, baik yang berbasis Microsoft maupun platform lainnya. Hal ini dicapai melalui penggunaan berbagai protokol komunikasi dan format data standar, seperti HTTP, REST, dan JSON, yang memungkinkan aplikasi untuk saling berinteraksi tanpa tergantung pada bahasa pemrograman tertentu.

Keunggulan dari .NET adalah kemampuannya untuk menjalankan aplikasi di berbagai lingkungan, termasuk Windows, Linux, dan macOS, berkat adanya .NET Core. Ini memberikan fleksibilitas yang lebih besar bagi organisasi untuk memilih infrastruktur dan platform yang paling sesuai dengan kebutuhan mereka. Selain itu, .NET juga menawarkan fitur-fitur canggih, seperti pengelolaan memori otomatis dan keamanan tingkat tinggi, yang memperkuat integritas dan stabilitas sistem.

Implementasi standar interaksi layanan ini juga mendorong penggunaan arsitektur berbasis layanan (*Service-Oriented Architecture*/SOA) dan microservices, di mana setiap layanan dapat dikembangkan, di-deploy, dan dikelola secara independen. Hal ini memungkinkan tim pengembangan untuk bekerja secara paralel pada bagian-bagian aplikasi yang berbeda, meningkatkan efisiensi dan mempercepat waktu ke pasar.

Standar interaksi layanan juga membantu dalam pengembangan dan pengelolaan API (*Application Programming Interface*) yang kuat. API yang dirancang dengan baik memungkinkan pihak ketiga untuk mengakses dan memanfaatkan layanan yang ada, membuka peluang untuk kolaborasi dan integrasi lebih lanjut dengan ekosistem yang lebih

luas. Secara keseluruhan, penerapan standar interaksi layanan melalui pemilihan teknologi seperti .NET tidak hanya memastikan bahwa sistem yang dibangun memiliki kemampuan untuk berkomunikasi dengan baik, tetapi juga memfasilitasi pengembangan yang lebih cepat, kolaborasi yang lebih baik, dan inovasi yang berkelanjutan dalam menghadapi kebutuhan bisnis yang terus berkembang. Dengan mematuhi standar ini, organisasi dapat menciptakan ekosistem layanan yang efisien dan responsif, siap untuk menghadapi tantangan dan peluang di masa depan.

Akses data adalah aspek krusial dalam arsitektur sistem yang mempengaruhi efisiensi, kecepatan, dan keamanan pengolahan informasi. Dalam konteks ini, terdapat beberapa pertimbangan penting yang harus diperhatikan, termasuk performa, standarisasi, arah masa depan, manajemen akses data, dan berbagai macam media penyimpanan data yang digunakan. Pertimbangan performa mencakup evaluasi kecepatan akses dan pengolahan data. Organisasi harus memastikan bahwa sistem yang dibangun mampu menangani beban kerja yang diharapkan, baik dalam hal jumlah transaksi yang diproses maupun dalam responsivitas sistem saat mengakses data. Penggunaan teknik seperti caching, pengindeksan, dan pemrograman paralel dapat membantu meningkatkan performa akses data, sehingga pengguna mendapatkan pengalaman yang lebih baik.

Standarisasi adalah penting untuk memastikan konsistensi dalam cara data diakses dan diolah. Mengadopsi standar terbuka untuk format data, protokol komunikasi, dan metode akses dapat memfasilitasi interoperabilitas antara sistem yang berbeda, mempermudah integrasi dan kolaborasi. Dengan adanya standarisasi, pengembang dapat dengan mudah mengadopsi teknologi baru tanpa harus memikirkan ulang cara akses data yang telah ada. Arah masa depan juga harus dipertimbangkan, terutama dengan pesatnya perkembangan teknologi seperti big data, cloud computing, dan Internet of Things (IoT). Organisasi perlu merencanakan akses data dengan memikirkan bagaimana tren dan inovasi baru dapat diakomodasi dalam arsitektur yang ada. Hal ini mencakup mempertimbangkan solusi yang skalabel dan fleksibel, yang dapat beradaptasi dengan perubahan kebutuhan bisnis dan teknologi.

Manajemen akses data adalah proses yang penting untuk menjaga keamanan dan integritas data. Ini mencakup kebijakan pengendalian akses yang menentukan siapa yang berhak mengakses data tertentu dan dalam konteks apa. Dengan menerapkan model kontrol akses yang baik, seperti *Role-Based Access Control* (RBAC) atau *Attribute-Based Access Control* (ABAC), organisasi dapat mengurangi risiko kebocoran data dan penyalahgunaan informasi sensitif. Selain itu, audit dan pemantauan akses juga harus dilakukan secara berkala untuk mendeteksi potensi pelanggaran dan memastikan kepatuhan terhadap kebijakan yang ditetapkan.

Macam media penyimpanan data yang digunakan juga memiliki dampak signifikan terhadap akses data. Pilihan antara penyimpanan lokal, cloud, atau hybrid harus dipertimbangkan dengan cermat. Setiap jenis media memiliki kelebihan dan kekurangan dalam hal biaya, kecepatan akses, kapasitas, dan keamanan. Misalnya, penyimpanan cloud menawarkan skalabilitas yang tinggi dan kemudahan akses dari mana saja, tetapi mungkin

menghadapi tantangan terkait latensi dan keamanan data. Di sisi lain, penyimpanan lokal mungkin menawarkan kecepatan akses yang lebih baik, tetapi dapat membatasi kemampuan organisasi untuk berbagi data secara efisien.

Dengan memperhatikan berbagai pertimbangan ini, organisasi dapat merancang sistem akses data yang tidak hanya efisien dan aman, tetapi juga mampu beradaptasi dengan perubahan di masa depan. Pendekatan yang holistik dan proaktif terhadap akses data akan memastikan bahwa organisasi dapat memanfaatkan informasi secara optimal, mendukung pengambilan keputusan yang lebih baik, dan meningkatkan daya saing di pasar.

Data storage merupakan komponen vital dalam infrastruktur teknologi informasi yang berfungsi untuk menyimpan semua informasi bisnis yang berkaitan dengan karyawan (employee), perusahaan (company), dan pelanggan (customer). Pengelolaan data yang efektif dan efisien sangat penting untuk mendukung operasional bisnis, pengambilan keputusan, dan strategi pertumbuhan jangka panjang. Penyimpanan data harus berbasis pada struktur yang konsisten dan terstandarisasi untuk memastikan integritas dan kemudahan akses data. Struktur data yang sama memungkinkan berbagai sistem dan aplikasi untuk memahami dan berinteraksi dengan data yang sama, mengurangi risiko kebingungan dan kesalahan. Hal ini juga berkontribusi pada kemampuan untuk mengintegrasikan berbagai sumber data dan menghasilkan analisis yang lebih komprehensif.

Lokasi informasi juga menjadi pertimbangan penting dalam desain data storage. Tempat penyimpanan data dapat memengaruhi performa sistem secara signifikan. Penyimpanan data yang terdistribusi di berbagai lokasi geografi dapat mempercepat akses bagi pengguna yang berada di lokasi tersebut, tetapi juga memerlukan sistem pengelolaan yang cermat untuk menjaga konsistensi dan sinkronisasi data. Misalnya, dengan menggunakan penyimpanan berbasis cloud, organisasi dapat memanfaatkan pusat data yang terletak di berbagai belahan dunia untuk memastikan akses yang cepat dan efisien, serta meningkatkan ketahanan dan redundansi.

Keputusan mengenai media penyimpanan juga sangat penting. Berbagai opsi, seperti *Hard Disk Drive* (HDD), *Solid-State Drive* (SSD), dan penyimpanan cloud, masing-masing memiliki kelebihan dan kekurangan. HDD umumnya lebih ekonomis untuk penyimpanan data dalam jumlah besar tetapi mungkin lebih lambat dalam kecepatan akses dibandingkan SSD, yang menawarkan kecepatan lebih tinggi tetapi dengan biaya yang lebih tinggi. Penyimpanan cloud memberikan fleksibilitas dan skalabilitas yang lebih baik, tetapi perlu diingat bahwa akses data melalui internet dapat mengalami latensi yang memengaruhi performa.

Selain itu, pengelolaan data backup dan pemulihan juga merupakan bagian dari strategi penyimpanan data. Membangun kebijakan yang jelas untuk pencadangan data dan pemulihan bencana sangat penting untuk melindungi informasi bisnis dari kehilangan akibat kegagalan sistem, bencana alam, atau serangan siber. Dengan memiliki rencana pemulihan yang solid, organisasi dapat memastikan bahwa data penting dapat dipulihkan dengan cepat, meminimalkan dampak terhadap operasional.

Secara keseluruhan, data storage yang baik harus dirancang dengan mempertimbangkan struktur yang konsisten, lokasi penyimpanan, performa sistem, serta

keamanan dan integritas data. Dengan pendekatan yang terencana dan sistematis, organisasi dapat mengelola informasi bisnis mereka secara efektif, mendukung pertumbuhan dan inovasi, serta mengurangi risiko yang terkait dengan pengelolaan data.

System services merupakan komponen penting dalam arsitektur teknologi informasi yang mencakup berbagai layanan yang disediakan oleh sistem untuk mendukung operasi bisnis dan kebutuhan pengguna. Evaluasi terhadap layanan sistem yang sedang digunakan adalah langkah awal yang krusial untuk memastikan bahwa layanan tersebut efektif, efisien, dan mampu memenuhi kebutuhan yang terus berkembang.

Dalam proses evaluasi ini, penting untuk menganalisis kinerja layanan yang ada, termasuk kecepatan, keandalan, dan tingkat kepuasan pengguna. Dengan melakukan analisis menyeluruh terhadap penggunaan layanan sistem, organisasi dapat mengidentifikasi kekuatan dan kelemahan dari layanan yang sedang digunakan. Misalnya, dengan melakukan survei pengguna atau analisis log sistem, organisasi dapat mengumpulkan umpan balik yang diperlukan untuk mengevaluasi apakah layanan tersebut memenuhi ekspektasi dan tujuan bisnis.

Selanjutnya, identifikasi teknologi yang mendukung layanan tersebut merupakan langkah penting untuk memastikan bahwa sistem memiliki fondasi yang tepat untuk operasional yang optimal. Teknologi yang digunakan harus sesuai dengan kebutuhan layanan, baik dari segi performa maupun kemampuan integrasi. Hal ini mencakup pemilihan platform perangkat keras dan perangkat lunak yang sesuai, serta protokol komunikasi yang efektif untuk memastikan layanan dapat berjalan dengan baik dan dapat diandalkan.

Penting juga untuk mempertimbangkan tendensi teknologi masa depan dalam identifikasi ini. Dengan kemajuan teknologi yang terus berkembang, seperti penggunaan kecerdasan buatan (AI), analitik data besar, dan cloud computing, organisasi harus bersiap untuk beradaptasi dan memanfaatkan inovasi baru yang dapat meningkatkan efisiensi dan efektivitas layanan sistem. Misalnya, penerapan layanan berbasis cloud dapat menawarkan fleksibilitas dan skalabilitas yang lebih baik, memungkinkan organisasi untuk mengatasi lonjakan permintaan dan mengoptimalkan biaya operasional.

Selain itu, organisasi harus melakukan penilaian risiko terhadap teknologi yang ada. Identifikasi potensi kerentanan dan tantangan yang mungkin muncul dari teknologi yang digunakan adalah kunci untuk menjaga keamanan dan stabilitas sistem. Dengan memiliki pemahaman yang jelas tentang risiko ini, organisasi dapat mengambil langkah-langkah proaktif untuk mengurangi dampaknya, baik melalui penguatan keamanan, pelatihan pengguna, atau penerapan kebijakan pemeliharaan yang tepat.

Secara keseluruhan, system services harus dievaluasi dan diperbarui secara berkala untuk memastikan bahwa mereka tetap relevan dan mendukung tujuan bisnis organisasi. Dengan mengevaluasi layanan sistem yang ada dan mengidentifikasi teknologi yang mendukungnya, organisasi dapat memastikan bahwa mereka tidak hanya dapat memenuhi kebutuhan saat ini tetapi juga siap untuk menghadapi tantangan dan peluang yang akan datang. Pendekatan yang terencana dalam pengelolaan system services akan membantu organisasi mencapai keunggulan kompetitif dan meningkatkan pengalaman pengguna.

Adapun tool pengembangan yang diperlukan untuk menciptakan, menguji, dan memelihara aplikasi dalam lingkungan teknologi informasi yang kompleks. Kemampuan untuk mengembangkan bagian-bagian dari aplikasi tidak hanya bergantung pada alat yang digunakan, tetapi juga pada pemahaman dan keterampilan tim pengembang dalam memanfaatkan alat tersebut secara efektif.

Pertama-tama, pemilihan tool pengembangan yang tepat sangat penting. Terdapat berbagai jenis alat yang dapat digunakan, mulai dari *Integrated Development Environments* (IDEs) seperti Visual Studio atau Eclipse, hingga alat manajemen versi seperti Git. IDE memungkinkan pengembang untuk menulis, menguji, dan debug kode dalam satu platform yang terintegrasi, sementara sistem manajemen versi memungkinkan kolaborasi tim yang lebih baik dan pengendalian versi yang lebih efisien.

Selain itu, framework pengembangan juga merupakan bagian penting dari toolset yang digunakan. Framework seperti .NET, Angular, atau React menyediakan struktur dasar dan komponen siap pakai yang mempercepat proses pengembangan dengan memungkinkan pengembang untuk fokus pada logika aplikasi, alih-alih membangun segala sesuatunya dari awal. Penggunaan framework yang sesuai tidak hanya meningkatkan efisiensi, tetapi juga membantu menjaga konsistensi dan kualitas kode.

Alat untuk pengujian dan pemantauan juga tidak kalah penting. Pengujian otomatisasi dengan alat seperti Selenium atau JUnit membantu memastikan bahwa aplikasi berfungsi dengan baik dan memenuhi spesifikasi yang ditetapkan. Dengan melakukan pengujian secara teratur selama siklus pengembangan, tim dapat mendeteksi dan memperbaiki masalah lebih awal, mengurangi biaya dan waktu yang dibutuhkan untuk perbaikan di kemudian hari. Selain itu, alat pemantauan seperti New Relic atau Splunk membantu dalam memantau kinerja aplikasi dan mendeteksi anomali yang mungkin mempengaruhi pengalaman pengguna.

Kolaborasi dalam tim juga menjadi kunci keberhasilan dalam penggunaan tool pengembangan. Alat seperti Jira untuk manajemen proyek dan Slack untuk komunikasi dapat membantu tim berkoordinasi dengan lebih baik, memastikan bahwa semua anggota tim selaras dalam tujuan pengembangan dan tenggat waktu. Kolaborasi yang baik meningkatkan efisiensi dan produktivitas, memungkinkan tim untuk lebih cepat merespons perubahan dan tantangan yang muncul.

Terakhir, pelatihan dan pengembangan keterampilan anggota tim juga merupakan aspek yang tidak boleh diabaikan. Meskipun alat yang baik dapat membantu dalam proses pengembangan, kemampuan individu dalam menggunakan alat tersebut secara efektif sangat menentukan keberhasilan proyek. Oleh karena itu, organisasi harus berinvestasi dalam pelatihan dan pengembangan keterampilan untuk memastikan bahwa tim pengembang tetap up-to-date dengan teknologi terbaru dan praktik terbaik dalam industri.

Sistem Operasi (SO) memainkan peran fundamental dalam pengembangan aplikasi dengan menyediakan lingkungan yang diperlukan untuk menjalankan perangkat lunak. Salah satu manfaat utama dari penggunaan sistem operasi adalah kemampuannya untuk mengurangi kebutuhan pengkodean (pemrograman) aplikasi berkat berbagai fitur dan layanan yang sudah tersedia.

Fitur-fitur dasar dalam sistem operasi, seperti manajemen memori, manajemen proses, dan sistem berkas, menyediakan fungsionalitas inti yang dapat dimanfaatkan oleh pengembang. Dengan memanfaatkan API (*Application Programming Interface*) yang disediakan oleh sistem operasi, pengembang dapat mengakses sumber daya sistem dan menjalankan fungsi dasar tanpa harus menulis kode dari nol. Misalnya, fungsi untuk membaca dan menulis data ke disk, mengelola input/output dari perangkat keras, atau melakukan operasi jaringan dapat dilakukan dengan lebih efisien menggunakan fungsi yang sudah disediakan oleh SO.

Selain itu, fitur keamanan yang disediakan oleh sistem operasi juga membantu dalam mengurangi kompleksitas pengembangan aplikasi. Sistem operasi modern biasanya dilengkapi dengan mekanisme keamanan, seperti kontrol akses, autentikasi, dan enkripsi, yang memungkinkan pengembang untuk fokus pada logika bisnis aplikasi mereka tanpa harus khawatir tentang implementasi detail keamanan secara menyeluruh. Dengan demikian, pengembang dapat mengandalkan keamanan yang diberikan oleh sistem operasi untuk melindungi data dan sumber daya aplikasi.

Kompatibilitas perangkat keras juga menjadi salah satu keuntungan dari sistem operasi. Dengan mendukung berbagai jenis perangkat keras, sistem operasi memungkinkan aplikasi yang dikembangkan dapat berjalan di berbagai konfigurasi sistem tanpa perlu menulis kode khusus untuk masing-masing perangkat. Hal ini sangat menguntungkan, terutama dalam konteks pengembangan aplikasi yang ditujukan untuk digunakan di berbagai platform dan perangkat.

Penggunaan framework dan pustaka yang terintegrasi dengan sistem operasi juga mempercepat proses pengembangan. Banyak sistem operasi menyediakan pustaka standar yang menawarkan berbagai fungsi tambahan, seperti grafik, jaringan, dan multimedia. Dengan menggunakan pustaka ini, pengembang dapat mengurangi jumlah kode yang perlu ditulis dan mempercepat proses pengembangan dengan memanfaatkan solusi yang telah teruji dan dioptimalkan.

Dukungan untuk pengembangan aplikasi berbasis web dan mobile juga menjadi bagian penting dari peran sistem operasi. Banyak sistem operasi modern mendukung platform pengembangan yang memungkinkan aplikasi dapat dijalankan di berbagai perangkat, baik desktop maupun mobile. Ini berarti bahwa pengembang dapat mengembangkan aplikasi yang responsif dan dapat diakses dari berbagai jenis perangkat dengan menggunakan teknologi yang sama, seperti HTML, CSS, dan JavaScript.

Secara keseluruhan, sistem operasi yang baik dapat secara signifikan mengurangi kebutuhan pengkodean aplikasi dengan menyediakan berbagai fitur dan layanan yang sudah ada. Dengan memanfaatkan kemampuan yang ditawarkan oleh sistem operasi, pengembang dapat fokus pada aspek yang lebih kritis dari aplikasi, seperti desain dan pengalaman pengguna, daripada terjebak dalam rincian teknis yang mendasari. Pendekatan ini tidak hanya meningkatkan efisiensi pengembangan tetapi juga memungkinkan pengembang untuk menciptakan aplikasi yang lebih inovatif dan relevan dengan kebutuhan pengguna.

Langkah selanjutnya adalah melakukan identifikasi terhadap objek-objek bisnis. Objek dapat berupa orang atau berbagai hal yang dideskripsikan dalam usage scenario. Objek-objek bisnis merupakan dasar bagi seluruh services (layanan), attributes (atribut), dan relationship (hubungan).

10.5 IDENTIFIKASI OBJEK BISNIS

Mengidentifikasi objek bisnis, layanan (service), dan atribut dalam pengembangan sistem informasi adalah langkah penting untuk memastikan bahwa aplikasi yang dibangun dapat memenuhi kebutuhan bisnis dengan baik. Adapun beberapa langkah yang bisa dilakukan untuk melakukan identifikasi ini yaitu menganalisis kebutuhan bisnis, pemetaan proses bisnis, identifikasi objek bisnis, identifikasi layanan, identifikasi service, dan uji serta memvalidasi. Objek bisnis dapat ditemukan di dalam diagram use case dan usage scenario, yang merupakan alat penting dalam mendefinisikan interaksi antara pengguna dan sistem.

Dalam konteks ini, objek bisnis dapat berupa orang, entitas, atau hal lainnya yang berperan dalam proses bisnis. Contohnya termasuk pekerja, pelanggan, kontrak, nomor pelanggan, produk, dan lainnya. Dalam diagram use case, objek bisnis biasanya diwakili sebagai aktor yang berinteraksi dengan sistem. Aktor ini dapat berupa pengguna akhir, seperti pelanggan yang menggunakan aplikasi untuk melakukan pembelian, atau pekerja yang menggunakan sistem untuk mengelola inventaris. Dengan mendefinisikan aktor-aktor ini, tim pengembang dapat lebih memahami kebutuhan dan harapan pengguna serta bagaimana sistem harus berfungsi untuk memenuhi kebutuhan tersebut.

Penggunaan scenario dalam mendeskripsikan penggunaan sistem secara lebih rinci juga memungkinkan untuk menggali lebih dalam objek bisnis. Misalnya, dalam sebuah usage scenario yang menggambarkan proses pembelian produk, objek bisnis yang terlibat mencakup pelanggan yang melakukan pembelian, produk yang dibeli, dan transaksi yang tercatat. Dengan memetakan alur cerita dari interaksi ini, tim dapat mengidentifikasi detail penting seperti atribut yang diperlukan untuk objek tersebut, serta bagaimana mereka saling berhubungan dalam konteks sistem. Selain itu, identifikasi objek bisnis juga membantu dalam pengembangan model data yang tepat. Setiap objek bisnis biasanya memiliki atribut yang relevan, seperti nama, alamat, dan ID yang unik.

Dengan mendokumentasikan objek dan atribut ini, tim dapat merancang basis data yang efisien dan sesuai dengan kebutuhan aplikasi. Misalnya, tabel dalam basis data dapat mencerminkan objek bisnis seperti pelanggan, dengan kolom yang sesuai untuk menyimpan informasi penting seperti nomor pelanggan, alamat, dan data kontak. Secara keseluruhan, pengidentifikasian objek bisnis yang tepat melalui diagram use case dan usage scenario merupakan langkah awal yang krusial dalam pengembangan sistem informasi. Dengan pemahaman yang jelas tentang siapa atau apa yang terlibat dalam sistem, serta interaksi yang terjadi di antara mereka, tim pengembang dapat menciptakan solusi yang lebih efektif dan responsif terhadap kebutuhan bisnis yang ada.

Pendekatan ini tidak hanya meningkatkan kualitas sistem yang dikembangkan, tetapi juga memastikan bahwa aplikasi yang dihasilkan relevan dan memberikan nilai tambah bagi

pengguna. Layanan (service) adalah aksi yang dilakukan oleh objek bisnis dan dapat ditemukan di dalam usage scenario, yang mendefinisikan bagaimana objek-objek tersebut berinteraksi satu sama lain untuk mencapai tujuan tertentu. Sebagai contoh, dalam sebuah skenario penggunaan, layanan yang umum mungkin termasuk menghitung total pembelian, menetapkan biaya pengiriman, dan memproses pembayaran. Setiap layanan merepresentasikan fungsi spesifik yang memberikan nilai tambah kepada pengguna atau sistem, dan sering kali berfokus pada hasil tertentu.

Misalnya, ketika pelanggan melakukan pembelian, layanan menghitung total pembelian berdasarkan item yang dipilih, termasuk memperhitungkan diskon, pajak, dan biaya pengiriman. Dalam hal ini, objek bisnis seperti "keranjang belanja" dan "pelanggan" berinteraksi melalui layanan ini untuk menghasilkan informasi yang dibutuhkan oleh pelanggan sebelum menyelesaikan transaksi. Identifikasi layanan dalam usage scenario juga membantu dalam merumuskan spesifikasi sistem yang jelas. Dengan mendeskripsikan setiap layanan, pengembang dapat mengidentifikasi input yang diperlukan, proses yang akan dilakukan, dan output yang diharapkan. Ini termasuk mendetailkan langkah-langkah yang harus diambil untuk menyelesaikan layanan tersebut dan bagaimana hasil akhir akan disajikan kepada pengguna.

Misalnya, dalam layanan penetapan biaya pengiriman, proses dapat mencakup pengambilan alamat pengiriman, menghitung jarak, dan menggunakan tarif pengiriman yang sesuai untuk memberikan estimasi biaya kepada pelanggan. Lebih lanjut, layanan juga dapat mencakup interaksi dengan layanan eksternal. Misalnya, dalam menghitung total pembelian, sistem mungkin perlu berinteraksi dengan layanan pembayaran atau sistem inventaris untuk memastikan ketersediaan produk. Dengan mengidentifikasi layanan ini, tim pengembang dapat merencanakan integrasi yang diperlukan dan memastikan bahwa sistem dapat berfungsi secara holistik. Penting untuk dicatat bahwa layanan tidak hanya terfokus pada tindakan yang bersifat teknis, tetapi juga mencakup interaksi yang lebih luas yang meningkatkan pengalaman pengguna. Misalnya, layanan pelanggan seperti memberikan dukungan, menjawab pertanyaan, atau memproses pengembalian barang juga merupakan bagian dari layanan yang dapat mendukung objek bisnis seperti pelanggan dan karyawan.

Atribut atau properti suatu objek adalah sebuah nilai data yang melekat pada objek, berfungsi untuk mendeskripsikan karakteristik atau informasi yang relevan tentang objek tersebut. Atribut memberikan konteks dan detail yang diperlukan untuk memahami objek dalam sistem, serta memainkan peran penting dalam pengelolaan data. Sebagai contoh, objek bisnis yang bernama Customer memiliki sejumlah atribut yang mendefinisikan identitas dan informasi penting mengenai pelanggan tersebut. Beberapa atribut utama dari objek Customer antara lain:

a. **No Akun**: Merupakan identifikasi unik yang diberikan kepada setiap pelanggan, berfungsi untuk membedakan satu pelanggan dengan pelanggan lainnya dalam sistem. Atribut ini biasanya berupa string atau angka yang memiliki format tertentu, dan menjadi kunci utama dalam basis data.

b. **Nama**: Merupakan nama lengkap pelanggan. Atribut ini penting untuk personalisasi komunikasi dan interaksi dengan pelanggan, serta berfungsi dalam berbagai proses bisnis, seperti pengiriman produk dan pembuatan faktur.
c. **Alamat**: Merupakan detail lokasi tempat tinggal atau pengiriman pelanggan. Atribut ini diperlukan untuk memastikan bahwa produk dapat dikirimkan dengan tepat dan untuk keperluan komunikasi lebih lanjut.
d. **No telepon**: Atribut ini memberikan informasi kontak yang diperlukan untuk menghubungi pelanggan. Ini penting dalam konteks layanan pelanggan, pengingat, dan komunikasi terkait pesanan.

Dalam konteks pengembangan sistem, atribut tidak hanya berfungsi sebagai pengidentifikasi, tetapi juga mempengaruhi bagaimana data disimpan, diakses, dan dikelola. Setiap atribut harus dirancang dengan mempertimbangkan tipe data, ukuran, dan validasi untuk memastikan integritas data. Misalnya, No Akun mungkin ditentukan sebagai integer dengan panjang tertentu, sedangkan Nama dan Alamat mungkin ditetapkan sebagai string dengan batasan panjang karakter tertentu. Lebih lanjut, atribut dapat dikategorikan menjadi atribut kunci dan atribut non-kunci. Atribut kunci, seperti No Akun, memainkan peran penting dalam mengidentifikasi objek secara unik dalam basis data.

Sebaliknya, atribut non-kunci memberikan informasi tambahan tetapi tidak diperlukan untuk pengenalan objek. Dalam desain basis data, pemahaman yang baik tentang atribut objek membantu dalam pembuatan tabel yang tepat dan relasi antar tabel. Dengan mendokumentasikan atribut secara sistematis, tim pengembang dapat merancang skema basis data yang efisien dan mampu menangani kebutuhan penyimpanan dan pengolahan data dalam aplikasi. Secara keseluruhan, atribut objek merupakan elemen penting dalam pengembangan sistem informasi. Mereka memberikan detail yang diperlukan untuk membangun aplikasi yang kaya fitur dan dapat memenuhi kebutuhan bisnis yang beragam, serta berkontribusi pada pengambilan keputusan yang lebih baik berdasarkan data yang ada. Dan yang terakhir adalah hubungan atau relationship. Hubungan atau relationship disini menggambarkan bagaimana sebuah objek terhubung dengan objek lain.

Identifikasi Objek

Usage scenario mungkin tidak secara langsung mengandung objek tertentu. Objek bisa tersembunyi dalam kalimat UC atau US tergantung bagaimana usage scenario ditulis. Berikut adalah beberapa contohnya:

a. Karyawan membuat kontrak dengan pelanggan
b. Karyawan membuat tindakan didalam sistem
c. Customer sebagai penerima tindakan yang dilakukan karyawan
d. Object bisnis = karyawan dan customer
e. Service = membuat kontrak
f. Atribut = nama customer, nama karyawan, dan lainnya
g. Relasi = karyawan melayani customer

Mengidentifikasi Layanan

Layanan adalah operasi, fungsi, atau transformasi yang dilakukan oleh objek dalam sistem. Dalam konteks pengembangan sistem informasi, layanan berfungsi sebagai jembatan yang menghubungkan interaksi antara pengguna dan sistem, serta memungkinkan objek untuk menjalankan tugas tertentu. Layanan dapat berperan dalam berbagai aspek fungsionalitas sistem, mulai dari pemrosesan data hingga komunikasi dengan komponen lain. Layanan memainkan peran sentral dalam mendefinisikan bagaimana objek beroperasi dan berinteraksi dalam sistem. Dengan merancang layanan yang efektif dan efisien, tim pengembang dapat memastikan bahwa aplikasi dapat memenuhi kebutuhan pengguna dengan baik, sambil juga memfasilitasi integrasi dan pengelolaan data yang optimal. Hal ini tidak hanya meningkatkan kinerja sistem, tetapi juga memberikan pengalaman pengguna yang lebih baik. Berikut adalah beberapa contoh layanan:

a. Menghitung jumlah total
b. Menetapkan biaya pengiriman
c. Melihat nama pelanggan

10.6 MENGIDENTIFIKASI RELASI

Menggambarkan bagaimana cara objek dihubungkan dengan objek lain adalah langkah penting dalam merancang sistem yang kompleks. Dalam *Unified Modeling Language* (UML), hubungan antar objek didefinisikan melalui berbagai tipe relasi, seperti dependensi, asosiasi, generalisasi, dan lainnya. Setiap tipe relasi memiliki karakteristik dan makna yang spesifik, yang membantu dalam memahami interaksi antar objek dalam sistem.

1. Dependensi: Ini adalah hubungan di mana perubahan pada satu objek dapat memengaruhi objek lain. Misalnya, jika seorang konsultan mengubah cara mereka menyediakan layanan, ini dapat berdampak pada pelanggan yang menerima layanan tersebut.
2. Asosiasi: Merupakan hubungan yang menunjukkan bahwa dua objek saling berinteraksi satu sama lain. Contohnya, Konsultan dan Customer berelasi karena mereka saling berinteraksi. Konsultan memberikan layanan kepada perusahaan, dan perusahaan tersebut memerlukan layanan dari konsultan. Hubungan ini dapat digambarkan dalam diagram UML dengan garis yang menghubungkan kedua objek, dan dapat dilengkapi dengan label yang menjelaskan sifat hubungan tersebut, seperti 'memberikan layanan' atau 'membutuhkan layanan'.
3. Generalisasi: Hubungan ini menunjukkan hierarki antara objek, di mana satu objek (superclass) dapat memiliki satu atau lebih objek turunan (subclass) yang mewarisi sifat dan perilakunya. Misalnya, objek Konsultan dapat digeneralisasi menjadi subkelas yang lebih spesifik, seperti Konsultan IT dan Konsultan Manajemen, masing-masing dengan atribut dan layanan yang lebih sesuai dengan spesialisasi mereka.

Dengan menggunakan relasi ini dalam desain, tim pengembang dapat memodelkan bagaimana objek berinteraksi dan bergantung satu sama lain. Ini tidak hanya membantu dalam merancang sistem yang lebih terstruktur, tetapi juga memudahkan pemeliharaan dan

pengembangan sistem di masa depan. Sebagai contoh, dalam konteks perusahaan yang menggunakan jasa konsultan, relasi antara Konsultan dan Customer menggambarkan dua peran penting dalam sistem.

Konsultan memberikan layanan yang spesifik, seperti analisis pasar atau pengembangan strategi bisnis, sementara pelanggan membutuhkan layanan tersebut untuk meningkatkan operasional atau mencapai tujuan bisnis mereka. Melalui pemahaman yang jelas mengenai hubungan ini, tim dapat merancang sistem yang lebih responsif terhadap kebutuhan pengguna, serta mengidentifikasi potensi area untuk perbaikan dalam proses interaksi. Selain itu, mendefinisikan hubungan antar objek dengan tepat juga berkontribusi pada pengembangan dokumentasi yang lebih baik, yang penting untuk kolaborasi tim dan pemeliharaan sistem di masa depan.

10.7 DOKUMENTASI DESAIN LOGIKAL

Dokumentasi desain logikal merupakan komponen penting dalam pengembangan sistem, yang menyajikan hasil dari proses desain logikal secara terstruktur. Hasil dari desain logikal ini biasanya mencakup beberapa elemen kunci, antara lain:

1. Model Objek: Ini menggambarkan objek-objek yang ada dalam sistem dan bagaimana mereka berinteraksi satu sama lain. Model objek menyajikan representasi visual dari objek dan relasi yang telah diidentifikasi, memungkinkan tim pengembang untuk memahami struktur dan fungsi sistem dengan lebih baik. Dalam model ini, objek akan mencakup atribut dan layanan yang telah didefinisikan sebelumnya, serta tipe relasi antara objek-objek tersebut, seperti asosiasi dan dependensi.
2. Model Data: Model data menggambarkan bagaimana data akan disimpan, diorganisasikan, dan diakses dalam sistem. Ini mencakup definisi tabel, atribut, tipe data, dan hubungan antar tabel dalam basis data. Model data sangat penting untuk memastikan integritas dan konsistensi data, serta memudahkan dalam pengambilan keputusan berbasis data. Dokumentasi ini membantu tim pengembang dalam merancang basis data yang efisien dan mendukung kebutuhan aplikasi.
3. High-Level User Interface (UI): Desain UI pada tingkat tinggi memberikan gambaran tentang antarmuka pengguna yang akan digunakan dalam sistem. Ini mencakup layout, elemen navigasi, dan interaksi pengguna yang direncanakan. Meskipun belum dalam bentuk detail, high-level UI membantu pemangku kepentingan untuk memahami bagaimana pengguna akan berinteraksi dengan sistem dan memastikan bahwa antarmuka tersebut memenuhi kebutuhan pengguna. Desain UI yang baik akan meningkatkan pengalaman pengguna dan mempermudah navigasi dalam aplikasi.

Dokumentasi desain logikal yang lengkap dan jelas sangat penting untuk memastikan bahwa semua pihak yang terlibat termasuk pengembang, desainer, dan pemangku kepentingan memiliki pemahaman yang sama mengenai struktur dan fungsi sistem yang akan dibangun. Selain itu, dokumentasi ini juga berfungsi sebagai referensi berharga selama fase pengembangan dan pemeliharaan sistem, serta membantu dalam pengujian dan validasi bahwa sistem telah dibangun sesuai dengan spesifikasi yang ditetapkan.

Metoda yang dapat dilakukan untuk mendokumentasikan dan merancang sistem dalam konteks desain logikal mencakup beberapa pendekatan yang efektif, dua di antaranya adalah *Class-Responsibility-Collaboration* (CRC) dan Sequence Diagram. Masing-masing metode ini memiliki tujuan dan fungsi spesifik yang dapat membantu tim dalam memahami dan merancang sistem dengan lebih baik.

1. *Class-Responsibility-Collaboration* (CRC): Metoda CRC adalah teknik yang digunakan untuk merancang kelas dalam sistem dengan cara yang sederhana dan kolaboratif. Dalam CRC, setiap kelas diidentifikasi bersama dengan tanggung jawab (responsibility) dan kolaborasi yang diperlukan untuk menjalankan fungsinya. Proses ini dilakukan dengan mengumpulkan anggota tim untuk mendiskusikan dan mendefinisikan setiap kelas, apa yang menjadi tanggung jawabnya, dan kelas-kelas lain yang akan berkolaborasi dengannya. Pendekatan ini membantu tim untuk berfokus pada interaksi antar kelas, serta memperjelas peran masing-masing dalam sistem, sehingga memudahkan pemahaman terhadap dinamika objek dalam konteks aplikasi.
2. Sequence Diagram: Diagram urutan (Sequence Diagram) adalah alat visual yang digunakan untuk menggambarkan interaksi antara objek dalam urutan waktu tertentu. Diagram ini menunjukkan bagaimana objek berkomunikasi satu sama lain melalui pesan, dan mengilustrasikan bagaimana skenario tertentu akan berlangsung dari waktu ke waktu. Dengan menggunakan Sequence Diagram, tim dapat memvisualisasikan alur proses, termasuk kapan dan bagaimana setiap objek berinteraksi, serta respons yang dihasilkan dari interaksi tersebut. Ini sangat berguna untuk memahami dan menganalisis alur kerja aplikasi, serta untuk mendeteksi potensi masalah dalam komunikasi antar objek.

Menggunakan kombinasi dari kedua metode ini, tim pengembang dapat menghasilkan desain yang lebih komprehensif dan terstruktur. CRC membantu dalam mendefinisikan dan mengorganisir kelas serta tanggung jawabnya, sementara Sequence Diagram memberikan perspektif yang jelas mengenai alur interaksi yang terjadi. Dengan pendekatan ini, proses desain logikal menjadi lebih terarah dan kolaboratif, memungkinkan semua anggota tim untuk berkontribusi dalam pengembangan sistem yang efisien dan efektif. Keduanya berperan penting dalam memastikan bahwa desain yang dihasilkan tidak hanya memenuhi kebutuhan fungsional, tetapi juga siap untuk menghadapi perubahan dan pengembangan di masa depan.

Kartu *Class-Responsibility-Collaboration* (CRC) membantu tim untuk fokus pada high-level responsibility dari setiap kelas dalam sistem. Dengan menggunakan kartu CRC, setiap kelas dapat diuraikan secara jelas dalam format yang sederhana, yang terdiri dari tiga komponen utama: nama kelas, tanggung jawab (responsibility), dan kolaborasi (collaboration).

1. Nama Kelas: Menyediakan identifikasi yang jelas untuk setiap kelas yang akan dirancang. Nama kelas biasanya merefleksikan entitas yang dimodelkan, seperti 'Customer', 'Order', atau 'Product', sehingga memudahkan anggota tim untuk memahami peran kelas tersebut dalam konteks aplikasi.

2. Tanggung Jawab (Responsibility): Merupakan daftar fungsi atau tugas yang harus dijalankan oleh kelas tersebut. Dengan mendefinisikan tanggung jawab secara eksplisit, tim dapat memastikan bahwa setiap kelas memiliki peran yang jelas dan tidak tumpang tindih dengan kelas lainnya. Misalnya, kelas 'Customer' mungkin memiliki tanggung jawab untuk menyimpan informasi pelanggan dan memproses pesanan.
3. Kolaborasi (Collaboration): Menggambarkan kelas lain yang berinteraksi dengan kelas tersebut untuk menyelesaikan tugas-tugas tertentu. Ini membantu tim untuk memahami bagaimana kelas berkomunikasi dan berinteraksi satu sama lain dalam menjalankan fungsi sistem secara keseluruhan. Misalnya, kelas 'Order' mungkin berkolaborasi dengan kelas 'Product' untuk menghitung total harga pesanan.

Dengan menyusun informasi ini dalam format kartu, tim pengembang dapat dengan cepat merujuk kembali ke tanggung jawab dan kolaborasi yang terkait dengan masing-masing kelas saat melakukan perancangan dan pengembangan. Kartu CRC juga berfungsi sebagai alat komunikasi yang efektif, memungkinkan anggota tim untuk berdiskusi tentang desain kelas dengan cara yang terstruktur dan mudah dipahami.

Selain itu, penggunaan kartu CRC dapat membantu dalam proses pengujian dan pemeliharaan sistem. Dengan memahami tanggung jawab dan interaksi antar kelas, tim dapat lebih mudah mengidentifikasi area yang perlu diuji secara mendalam atau yang mungkin memerlukan perhatian khusus saat ada perubahan dalam desain. Secara keseluruhan, kartu CRC memperkuat fokus pada tanggung jawab kelas yang tinggi, memastikan bahwa semua aspek penting dari desain kelas diperhatikan, dan mendukung kolaborasi yang efektif dalam tim pengembangan.

10.8 DESAIN FISIKAL

Physical design adalah tahap penting dalam proses pengembangan perangkat lunak yang berfokus pada penerapan desain logikal ke dalam komponen fisik, layanan, dan teknologi yang diperlukan untuk membangun solusi. Proses ini menguraikan bagaimana model logikal akan direalisasikan dalam dunia nyata dengan mempertimbangkan kebutuhan spesifik pengembangan perangkat lunak. Dalam konteks ini, physical design tidak hanya mencakup struktur komponen dan layanan, tetapi juga mencakup batasan teknologi yang ada, sehingga solusi yang dihasilkan dapat berfungsi secara efektif dalam lingkungan yang telah ditentukan.

Physical design bukan hanya langkah awal menuju implementasi, tetapi juga merupakan fondasi penting yang memastikan solusi perangkat lunak dapat beroperasi dalam lingkungan nyata dengan memenuhi kebutuhan bisnis yang spesifik. Dengan menyusun setiap komponen dan layanan secara detail pada tahap ini, tim pengembang dapat mengantisipasi potensi hambatan teknis serta memastikan bahwa setiap elemen dalam sistem akan bekerja dengan harmonis untuk mencapai tujuan organisasi. Pada tahap ini, beberapa aspek yang diperhatikan meliputi:

1. Penguraian Komponen: Di sini, setiap komponen sistem diidentifikasi dan dijelaskan secara mendetail, termasuk bagaimana mereka akan berfungsi secara independen dan berinteraksi satu sama lain. Misalnya, komponen seperti server, basis data, dan

antarmuka pengguna diuraikan untuk memastikan bahwa semua elemen sistem berfungsi secara harmonis.

2. Layanan yang Diberikan: Layanan yang disediakan oleh setiap komponen juga didefinisikan dengan jelas. Ini mencakup layanan yang harus dijalankan, serta cara mereka berinteraksi dengan komponen lain dalam sistem. Dengan cara ini, tim dapat memastikan bahwa layanan tersebut memenuhi kebutuhan bisnis dan pengguna akhir.
3. Teknologi yang Digunakan: Physical design juga melibatkan pemilihan teknologi yang akan digunakan untuk membangun sistem. Ini mencakup tool pengembangan yang akan digunakan, seperti framework dan bahasa pemrograman, serta lingkungan penyebaran solusi, seperti server dan platform cloud. Pemilihan teknologi ini sangat penting karena dapat mempengaruhi performa, skalabilitas, dan pemeliharaan sistem di masa depan.

Input informasi ke dalam physical design meliputi beberapa elemen penting, yaitu:

1. Model Objek Logikal: Menyediakan struktur dan definisi objek yang akan digunakan dalam sistem, beserta tanggung jawab dan interaksinya.
2. Desain Level Tinggi User Interface: Memberikan gambaran tentang antarmuka pengguna, termasuk layout dan elemen navigasi, yang akan digunakan dalam aplikasi. Desain ini membantu memastikan bahwa antarmuka pengguna sesuai dengan pengalaman pengguna yang diinginkan.
3. Model Data Logikal: Menyediakan struktur data yang diperlukan untuk menyimpan informasi dalam sistem, termasuk hubungan antara data dan cara data akan diakses. Model data ini sangat penting untuk memastikan integritas dan efisiensi pengelolaan data.

Dengan mengintegrasikan semua elemen ini, physical design memungkinkan tim untuk menghasilkan rencana yang komprehensif untuk implementasi solusi. Hal ini juga memberikan fondasi yang kuat untuk pengujian dan validasi sistem, memastikan bahwa solusi yang dibangun tidak hanya memenuhi spesifikasi teknis, tetapi juga sesuai dengan kebutuhan bisnis dan pengguna. Pada akhirnya, tahap physical design merupakan jembatan antara desain konseptual dan realisasi teknis, yang sangat penting untuk keberhasilan proyek pengembangan perangkat lunak.

Scope Pyschal Design

Physical design bukanlah proses pembuatan kode program, tetapi lebih pada membuat spesifikasi detail dari komponen aplikasi yang akan digunakan pada tahap pengembangan. Pada tahap ini, physical design berfokus pada dokumentasi rinci mengenai bagaimana setiap komponen dalam sistem akan berfungsi, terhubung, dan berinteraksi, serta di mana komponen-komponen tersebut akan dialokasikan dalam infrastruktur yang ada. Beberapa aktivitas utama yang dilakukan dalam physical design meliputi:

1. Pembuatan Spesifikasi Detail Komponen: Physical design menyediakan deskripsi teknis yang mendalam mengenai masing-masing komponen, seperti layanan yang akan diberikan, protokol komunikasi antar komponen, serta teknologi pendukung yang diperlukan. Spesifikasi ini membantu tim pengembang untuk memahami secara

mendalam setiap bagian sistem dan bagaimana mereka bekerja sama untuk menciptakan solusi yang efisien.

2. Menentukan Lokasi Komponen dalam Infrastruktur**: Physical design juga menentukan di mana setiap komponen aplikasi akan ditempatkan. Ini mencakup alokasi komponen di berbagai server, database, atau perangkat keras lain yang sesuai dengan lingkungan implementasi yang direncanakan. Alokasi ini memastikan bahwa setiap bagian dari sistem dapat beroperasi dengan kinerja optimal dan sesuai dengan kebutuhan skala dan keamanan yang ditetapkan.
3. Identifikasi Teknologi yang Sesuai: Meskipun bukan merupakan tahap penyebaran atau instalasi teknologi, physical design sangat berperan dalam **mengidentifikasi teknologi** yang paling sesuai untuk digunakan dalam pengembangan solusi. Ini mencakup pilihan terhadap platform, framework, tool pengembangan, sistem operasi, serta layanan eksternal yang mungkin dibutuhkan. Dengan pendekatan ini, physical design membantu memastikan bahwa teknologi yang dipilih mendukung tujuan jangka panjang sistem, termasuk skalabilitas, performa, dan kemudahan pemeliharaan.
4. Pemilihan Lingkungan Pengembangan: Physical design juga berperan dalam memilih lingkungan pengembangan yang akan digunakan oleh tim. Ini mencakup peralatan pengembangan, sistem manajemen versi, dan infrastruktur testing yang diperlukan untuk mendukung pengembangan perangkat lunak dengan efisien. Pilihan ini sangat penting untuk memperlancar proses pengembangan serta memastikan konsistensi dan kualitas pada setiap tahap pengembangan.

Physical design adalah tahap penting yang menjembatani antara desain logikal dan implementasi teknis. Dengan menyediakan spesifikasi teknis yang mendalam dan memilih teknologi yang sesuai, physical design memberikan panduan yang jelas bagi pengembang dalam membangun solusi yang dapat berfungsi dengan baik dalam lingkungan produksi. Hasil dari tahap physical design ini menjadi referensi yang kritis selama tahap pengembangan, pengujian, dan bahkan pemeliharaan, sehingga sistem yang dihasilkan dapat beroperasi sesuai ekspektasi dan siap untuk skala yang lebih besar di masa depan.

Tujuan Akhir Pyschal Desain

Tujuan akhir dari physical design adalah untuk menciptakan fondasi yang kuat dalam pengembangan aplikasi dengan berfokus pada identifikasi dan pemilihan teknologi yang paling sesuai, transformasi model logikal ke dalam model fisikal, serta penyediaan dasar untuk proses pengembangan yang berkelanjutan. Physical design bertujuan untuk mendetailkan aspek-aspek teknis dari desain yang telah ditetapkan, sehingga menjadi panduan bagi tim pengembang dalam melangkah ke tahap implementasi. Beberapa tujuan utama dari physical design meliputi:

1. Mengidentifikasi Teknologi yang Sesuai untuk Pengembangan Aplikasi: Tahap ini sangat penting karena memungkinkan tim untuk menentukan perangkat, platform, framework, dan tool pengembangan yang mendukung keberhasilan proyek. Pemilihan teknologi yang tepat membantu dalam memastikan bahwa aplikasi dapat beroperasi dengan efisien, skalabel, dan mudah untuk di-maintenance.

2. Mentranformasi Model Desain Logikal ke dalam Model Desain Fisikal: Dengan physical design, model konseptual yang sebelumnya hanya berupa desain logikal diterjemahkan ke dalam komponen fisik, layanan, dan teknologi yang akan diimplementasikan. Transformasi ini memungkinkan setiap komponen logikal mendapatkan bentuk nyata dan dapat diukur dalam lingkungan teknis.
3. Menyediakan Dasar untuk Proses Pengembangan Selanjutnya: Physical design menyediakan spesifikasi teknis yang mendalam, yang nantinya akan menjadi referensi utama bagi tim pengembang selama proses pengembangan berlangsung. Spesifikasi ini memastikan setiap komponen sistem memiliki tujuan dan metode implementasi yang jelas, serta memudahkan pengembang dalam menjalankan tahap coding, testing, dan deployment.
4. Mendefinisikan Milestone dan Persetujuan Rencana Proyek: Physical design juga menetapkan milestone-milestone penting dalam proses pengembangan aplikasi. Dengan adanya milestone, setiap tahapan dari proses desain dan pengembangan dapat dimonitor dan dievaluasi keberhasilannya sesuai jadwal yang telah ditentukan. Milestone ini membantu menjaga agar tim tetap berada pada jalur yang benar dan memastikan bahwa setiap tahap diselesaikan dengan persetujuan dari stakeholder yang terkait.

Deliverables Design Fisik

Deliverables dari physical design adalah dokumen dan diagram teknis yang sangat penting sebagai panduan lengkap untuk mengarahkan proses implementasi sistem. Semua deliverables ini memastikan bahwa sistem yang dikembangkan sesuai dengan spesifikasi teknis dan siap untuk diimplementasikan di lingkungan produksi. Berikut adalah deliverables utama yang dihasilkan dari physical design beserta detail kegunaannya.

1. Class Diagram: Class diagram memetakan kelas-kelas dalam sistem, termasuk atribut dan metode yang dimiliki oleh setiap kelas serta relasi yang ada di antara kelas-kelas tersebut, seperti inheritance, asosiasi, dan agregasi. Diagram ini berfungsi sebagai cetak biru untuk struktur dan interaksi objek dalam solusi yang akan dikembangkan, membantu tim memahami hierarki dan logika objek secara keseluruhan.
2. Model Komponen: Model komponen adalah representasi visual dari bagian-bagian modular dari sistem yang menunjukkan bagaimana masing-masing komponen aplikasi saling berinteraksi dan berkomunikasi. Model ini biasanya digunakan untuk memvisualisasikan antarmuka, dependensi, dan protokol komunikasi antar komponen. Model komponen ini juga membantu dalam mendefinisikan arsitektur solusi, baik dalam konteks komponen yang berbasis server, layanan microservices, maupun komponen lokal.
3. Sequence Diagram atau Activity Diagram: Sequence diagram menunjukkan urutan interaksi antara objek dalam suatu skenario tertentu, mendetailkan pesan yang dikirim dan diterima antar objek untuk menyelesaikan proses bisnis. Sequence diagram membantu dalam memahami alur waktu dari interaksi antar objek. Activity diagram memvisualisasikan aliran proses dari satu aktivitas ke aktivitas lain dalam sistem,

menunjukkan transisi antara tahap-tahap dan mendetailkan logika proses yang mendasari sistem. Diagram ini juga bermanfaat untuk menggambarkan kondisi pengambilan keputusan dalam alur proses.

4. Skema Database: Skema database mendefinisikan struktur penyimpanan data yang digunakan oleh aplikasi, termasuk tabel, kolom, indeks, constraints, serta relasi antar tabel. Skema ini dirancang untuk memastikan konsistensi dan efisiensi pengelolaan data dalam sistem. Skema database membantu database administrator dan pengembang backend dalam memahami struktur, mengoptimalkan kinerja, dan menjaga integritas data.
5. Topologi Jaringan: Topologi jaringan menjelaskan lokasi perangkat keras serta pola konektivitas jaringan. Ini mencakup distribusi server, client, router, switch, firewall, serta penempatan data center dan cloud. Topologi ini menjadi acuan untuk menjaga performa, keamanan, dan skala dari infrastruktur jaringan yang akan digunakan oleh solusi yang dikembangkan.
6. Topologi Data dan Komponen: Topologi data dan komponen mendefinisikan lokasi fisik dari komponen-komponen solusi, layanan, serta media penyimpanan data. Topologi ini membantu menentukan posisi dan interaksi antar bagian dari sistem di lingkungan produksi dan memastikan semua elemen terdistribusi dengan efisien dan aman sesuai kebutuhan performa dan skala aplikasi.
7. Spesifikasi Komponen Aplikasi: Spesifikasi ini mencakup detail teknis dari komponen aplikasi yang digunakan, seperti kapasitas server, kebutuhan perangkat keras (hardware), dan perangkat lunak (software) tambahan. Spesifikasi ini dirancang untuk memastikan bahwa semua komponen yang terlibat, baik aplikasi maupun infrastruktur, memiliki dukungan teknis yang sesuai untuk menjalankan fungsinya.
8. Strategi Pemaketan dan Distribusi Aplikasi: Strategi ini merinci cara aplikasi akan dipaketkan dan didistribusikan. Contohnya, apakah aplikasi akan disediakan dalam bentuk kontainer (misalnya Docker), paket installer, atau melalui sistem Continuous Deployment. Strategi ini juga menetapkan mekanisme untuk penyebaran dan pembaruan di masa mendatang, menjaga agar sistem dapat ditingkatkan tanpa mengganggu operasional yang sedang berjalan.
9. Model Pemrograman Aplikasi: Model ini mendefinisikan arsitektur aplikasi yang akan digunakan, seperti model client-server, microservices, atau aplikasi berbasis cloud. Model pemrograman aplikasi berfungsi sebagai panduan dalam menerapkan pola pengembangan yang paling sesuai dengan skala, performa, dan tujuan bisnis jangka panjang.

Dengan semua deliverables ini, hasil physical design menyediakan dokumen dan panduan teknis yang sangat penting untuk keberhasilan pengembangan dan penyebaran sistem. Semua deliverables ini tidak hanya menjadi referensi bagi tim pengembang, tetapi juga menjadi alat komunikasi yang memastikan bahwa setiap pihak terkait memahami struktur, logika, dan operasional dari sistem yang akan dibangun. Physical design juga membantu memastikan

bahwa setiap langkah dalam pengembangan telah terencana dengan baik sehingga risiko keterlambatan dan kesalahan dapat diminimalkan.

Analisi Desain Fisikal

Analisis Desain Fisikal merupakan tahap penting dalam proses pengembangan, di mana tim melakukan pemetaan dan penyaringan model desain fisik berdasarkan dokumentasi desain logikal yang telah disusun sebelumnya. Tahap ini dilakukan untuk memastikan bahwa setiap elemen dari desain logikal dapat diterjemahkan ke dalam spesifikasi teknis yang mendukung pengembangan perangkat lunak. Berikut adalah langkah-langkah utama dalam analisis desain fisikal:

1. Menyaring Desain Logikal Model UML: Pada tahap ini, tim memverifikasi model UML yang dihasilkan dari desain logikal, seperti class diagram, sequence diagram, dan komponen lain. Penyaringan ini mencakup pemastian bahwa semua relasi, atribut, layanan, dan skenario telah terdefinisi dengan jelas dan sesuai dengan kebutuhan fungsional serta non-fungsional sistem. Tim juga mengevaluasi ketepatan setiap elemen UML untuk memastikan bahwa tidak ada konflik atau kekurangan dalam model logikal yang bisa menghambat desain fisik dan pengembangan lebih lanjut.
2. Membuat Persiapan Model Pengembangan: Setelah penyaringan model logikal, tim memulai persiapan model pengembangan fisik. Ini mencakup identifikasi teknologi dan platform yang akan digunakan, seperti pemilihan framework pemrograman, database management system (DBMS), dan tools pengembangan lainnya. Model pengembangan ini juga menetapkan struktur pengembangan aplikasi, seperti pembagian modul berdasarkan komponen, pengelompokan layanan, dan penempatan data storage sesuai kebutuhan performa dan skalabilitas.
3. Menentukan Struktur Penyebaran Komponen: Pada analisis fisik, tim mengidentifikasi bagaimana setiap komponen aplikasi akan ditempatkan dalam infrastruktur jaringan. Ini mencakup pembagian aplikasi ke dalam lapisan front-end, back-end, dan database yang berinteraksi satu sama lain di dalam topologi jaringan yang direncanakan. Struktur penyebaran ini memastikan bahwa setiap komponen dapat diakses sesuai dengan kebutuhan operasional dan mengoptimalkan performa aplikasi di lingkungan pengguna.
4. Evaluasi Kesesuaian Teknologi: Tim melakukan evaluasi terhadap teknologi yang direncanakan untuk memastikan kesesuaian dengan persyaratan bisnis dan teknis. Ini mencakup evaluasi platform, compatibility antar teknologi, serta dukungan jangka panjang dari penyedia teknologi. Analisis ini juga mempertimbangkan aspek keamanan, integrasi dengan sistem lain, dan dukungan untuk pengembangan lanjutan di masa depan.
5. Pengembangan Dokumentasi Detail: Selama proses analisis desain fisikal, tim mengembangkan dokumentasi teknis yang lebih rinci yang mencakup spesifikasi setiap komponen, strategi deployment, dan parameter performa yang diharapkan. Dokumentasi ini akan menjadi acuan bagi pengembang saat mengimplementasikan aplikasi dan juga sebagai panduan bagi tim operasi saat sistem sudah di-deploy.

Dokumentasi ini juga memuat konfigurasi infrastruktur yang dibutuhkan, seperti kebutuhan server, database, jaringan, dan konfigurasi middleware yang akan digunakan.

Dengan melakukan analisis desain fisikal ini, tim dapat memastikan bahwa setiap aspek dari desain logikal telah dipertimbangkan dan diterjemahkan ke dalam model fisikal yang layak dan siap untuk proses pengembangan. Tahap ini penting agar semua keputusan teknis yang dibuat selaras dengan tujuan proyek dan menghasilkan solusi yang dapat diandalkan serta efisien dalam implementasinya.

Menyaring Model Uml

Pada akhir tahap desain logikal, tim menghasilkan model-model UML sebagai representasi visual dari struktur dan interaksi komponen dalam sistem. Model-model UML ini berperan penting sebagai dasar desain yang akan diterjemahkan dalam tahap desain fisik dan pengembangan. Berikut adalah model UML yang dihasilkan beserta fungsinya:

1. Objek dan Service Inventory: Objek dan service inventory adalah daftar yang mencakup semua objek bisnis dan layanan yang terlibat dalam sistem, serta mendokumentasikan atribut dan operasi (service) yang dimiliki setiap objek. Inventori ini membantu tim memahami komponen-komponen utama dan interaksi antar komponen dalam sistem, serta mempermudah identifikasi fitur-fitur yang dibutuhkan.
2. Class Diagram: Class diagram menggambarkan struktur kelas yang ada dalam sistem, termasuk atribut, metode, dan hubungan antar kelas, seperti inheritance, asosiasi, dan agregasi. Class diagram memberikan pemahaman mendalam tentang organisasi dan hierarki objek di dalam sistem, serta memungkinkan tim untuk merancang kelas dengan tanggung jawab spesifik.
3. Sequence Diagram: Sequence diagram menunjukkan urutan interaksi antar objek dalam suatu skenario tertentu untuk mencapai tujuan tertentu. Diagram ini memperlihatkan alur pesan dan panggilan metode yang terjadi dari satu objek ke objek lain sesuai dengan urutan waktu, yang penting untuk memahami alur eksekusi proses dan memastikan setiap interaksi antar komponen terdefinisi dengan baik.
4. Activity Diagram: Activity diagram menggambarkan aliran proses bisnis atau aliran aktivitas di dalam sistem, yang mencakup transisi dari satu aktivitas ke aktivitas berikutnya. Diagram ini berfungsi untuk mendokumentasikan alur logika proses, termasuk kondisi pengambilan keputusan dan cabang-cabang alur proses, sehingga memudahkan dalam perancangan logika yang kompleks.
5. Component Diagram: Component diagram menunjukkan bagaimana komponen-komponen utama dalam sistem, seperti modul-modul aplikasi, layanan, atau antarmuka, saling terhubung dan berinteraksi satu sama lain. Component diagram ini membantu tim memahami struktur modular dari sistem dan mendefinisikan dependensi antar komponen, yang penting dalam memastikan interoperabilitas dan skalabilitas sistem.

Keseluruhan model UML ini menjadi dasar untuk pengembangan lebih lanjut dalam desain fisik, di mana setiap komponen yang dirancang akan diterapkan dalam lingkungan teknologi

nyata. Model-model tersebut memastikan bahwa semua elemen desain sistem telah diuraikan secara rinci dan siap untuk proses implementasi.

Persiapan Model Penyebaran

Persiapan model penyebaran dalam desain fisik adalah proses yang memastikan setiap komponen aplikasi dapat berfungsi dengan optimal dalam lingkungan teknologi yang akan digunakan. Persiapan ini terdiri dari beberapa aspek penting, yaitu jaringan, data, dan topologi komponen, yang masing-masing berfungsi untuk menjamin bahwa sistem yang dikembangkan dapat diimplementasikan secara efektif dalam arsitektur infrastruktur yang direncanakan. Berikut adalah penjelasan masing-masing aspek:

1. Jaringan:
   a. Persiapan jaringan mencakup pengaturan infrastruktur jaringan yang mendukung komunikasi antar komponen sistem. Ini meliputi konfigurasi server, router, dan pengaturan firewall untuk memastikan keamanan serta kecepatan akses yang optimal. Persiapan ini juga mempertimbangkan load balancing, redundansi, dan failover, yang sangat penting dalam menjaga ketersediaan dan keandalan sistem.
   b. Selain itu, aspek jaringan juga mempertimbangkan lokasi fisik dari server, baik lokal maupun cloud, serta jaringan virtual yang diperlukan untuk mendukung komunikasi antar layanan dalam lingkungan terdistribusi.
2. Data:
   a. Persiapan data mencakup struktur penyimpanan data, alokasi database, dan pengaturan akses yang diperlukan untuk memastikan data dapat diakses dan dikelola dengan efisien. Persiapan ini juga meliputi pemilihan jenis database yang sesuai (misalnya SQL atau NoSQL) berdasarkan kebutuhan performa dan skala aplikasi, serta kebijakan backup dan recovery data.
   b. Data yang disimpan dalam sistem direncanakan untuk mendukung integrasi yang efisien dengan sistem lain, serta pengaturan hak akses untuk memastikan bahwa hanya pengguna yang berwenang yang dapat mengakses data tertentu sesuai dengan perannya.
3. Topologi Komponen:
   a. Topologi komponen menggambarkan bagaimana setiap komponen aplikasi, seperti modul layanan dan antarmuka pengguna, dihubungkan dalam struktur sistem secara keseluruhan. Topologi ini menentukan di mana setiap komponen ditempatkan dalam jaringan, apakah di sisi klien, server, atau layer middleware, untuk memastikan kelancaran komunikasi dan pemrosesan data antar komponen.
   b. Selain itu, topologi komponen mempertimbangkan skenario-skenario penskalaan (scaling), baik vertikal maupun horizontal, untuk mengakomodasi pertumbuhan jumlah pengguna atau data. Struktur komponen yang efektif memungkinkan fleksibilitas dan skalabilitas, sehingga aplikasi dapat dikembangkan lebih lanjut tanpa mengganggu arsitektur yang sudah ada.

Dengan persiapan model penyebaran yang matang, setiap komponen dari desain fisik dapat diimplementasikan dengan baik di dalam lingkungan teknologi yang telah dipilih, mendukung

performa, keamanan, dan skalabilitas sistem secara keseluruhan. Model penyebaran ini juga memastikan bahwa sistem siap untuk menjalani uji coba dan implementasi pada tahap selanjutnya.

10.9 MELENGKAPI TAHAPAN PLANNING

Perencanaan Fitur Administratif

Fitur administratif adalah komponen penting dalam mendukung pengelolaan, pemantauan, dan pemeliharaan solusi perangkat lunak. Fitur ini memungkinkan administrator untuk memantau kinerja sistem, mengelola data, serta memastikan kepatuhan terhadap perjanjian lisensi yang telah ditentukan. Dalam perencanaannya, fitur administratif terbagi ke dalam tiga jenis utama, yaitu:

1. Monitoring: Monitoring adalah fitur yang dirancang untuk mengawasi kinerja dan kondisi sistem secara real-time. Dengan fitur ini, administrator dapat memantau penggunaan sumber daya (seperti CPU, memori, dan kapasitas penyimpanan), beban jaringan, serta kesehatan aplikasi secara keseluruhan. Fitur ini memungkinkan identifikasi dini terhadap potensi masalah, seperti penggunaan sumber daya yang berlebihan atau kegagalan sistem, sehingga dapat dilakukan tindakan perbaikan sebelum terjadi gangguan operasional. Monitoring juga dapat mencakup pelacakan log aktivitas, yang memungkinkan administrator untuk melihat riwayat tindakan pengguna dan mencatat akses ke data penting. Ini adalah langkah penting dalam menjaga keamanan dan memastikan sistem berfungsi sesuai dengan standar keamanan yang telah ditetapkan.
2. Data Migration: Data migration adalah fitur yang memungkinkan pemindahan data dari sistem lama ke sistem baru, atau dari satu database ke database lain, baik secara permanen maupun temporer. Proses ini mencakup alat-alat yang membantu mempersiapkan data untuk migrasi, memastikan format dan kualitas data tetap terjaga selama proses berlangsung. Data migration menjadi penting dalam transisi ke sistem baru atau dalam pembaruan sistem yang memerlukan transfer data antar platform. Selain itu, fitur data migration juga mencakup pengelolaan data backup dan recovery. Ini memastikan bahwa data dapat dipulihkan ke kondisi semula jika terjadi kegagalan sistem, yang sangat penting dalam menjaga integritas data dan keandalan sistem.
3. Licensing Specification: Licensing specification adalah fitur administratif yang mengatur persyaratan lisensi untuk setiap komponen perangkat lunak yang digunakan dalam solusi. Fitur ini memungkinkan administrator untuk melacak status lisensi, kapan lisensi akan berakhir, serta memastikan bahwa perangkat lunak yang digunakan sesuai dengan jumlah lisensi yang telah disetujui, baik untuk penggunaan internal maupun eksternal. Selain itu, licensing specification juga mendukung proses pembaruan dan peningkatan lisensi jika kebutuhan sistem meningkat. Ini mencegah pelanggaran lisensi dan memastikan bahwa organisasi mematuhi ketentuan penggunaan yang ditetapkan oleh penyedia perangkat lunak. Sistem ini biasanya juga

mengintegrasikan notifikasi peringatan jika ada lisensi yang akan segera berakhir atau perlu diperpanjang.

Fitur administratif ini memastikan solusi perangkat lunak dapat dikelola dan dioperasikan dengan optimal, serta mendukung kepatuhan organisasi terhadap regulasi penggunaan sistem. Dengan perencanaan yang baik pada fitur administratif, perusahaan dapat menjaga kinerja dan keandalan sistem, memastikan integritas data, serta menghindari risiko pelanggaran lisensi yang dapat mengakibatkan sanksi atau penalti.

Monitoring digunakan untuk memastikan bahwa aplikasi berfungsi secara tepat dan berada pada kondisi optimal dengan memantau performa, ketersediaan, serta kondisi sumber daya yang digunakan. Fitur ini memungkinkan pengawasan real-time atas berbagai metrik kinerja, seperti penggunaan CPU, memori, bandwidth jaringan, dan kapasitas penyimpanan, untuk memastikan aplikasi tetap berjalan sesuai ekspektasi.

Automated monitoring, yang mencakup pemantauan otomatis melalui berbagai sistem dan alat-alat analitik, memberikan kemampuan untuk mengidentifikasi kondisi yang tidak normal atau anomali yang mungkin menandakan adanya permasalahan. Dengan deteksi otomatis, sistem dapat mengirimkan notifikasi atau peringatan kepada administrator untuk segera melakukan tindakan preventif atau perbaikan yang diperlukan. Hal ini sangat penting untuk mengurangi downtime, menghindari gangguan besar pada operasional, dan meminimalisir dampak pada pengguna akhir.

Sebagai contoh, sistem monitoring dapat mengirimkan peringatan jika ada lonjakan penggunaan CPU atau jika kapasitas penyimpanan mendekati batas maksimum. Dengan begitu, tim administrasi dapat merespons sebelum permasalahan menjadi serius. Monitoring juga memungkinkan tim untuk melacak riwayat performa dan analisis tren, sehingga membantu dalam perencanaan kapasitas serta identifikasi kebutuhan peningkatan infrastruktur di masa depan.

Dengan adanya automated monitoring, waktu yang dibutuhkan untuk mengembalikan kondisi sistem setelah terjadi kesalahan atau kerusakan dapat ditekan, dan solusi untuk masalah yang serupa di masa mendatang dapat disiapkan dengan lebih baik. Fitur monitoring ini, baik manual maupun otomatis, pada akhirnya mendukung kinerja sistem yang lebih handal dan responsif, serta menjamin ketersediaan aplikasi sesuai dengan kebutuhan operasional dan kepentingan pengguna.

Rencana monitoring berisi tiga komponen utama yang memastikan proses pemantauan berlangsung efektif dan memberikan nilai tambah bagi organisasi:

1. Apa saja yang akan dimonitor: Dalam bagian ini, tim harus mendefinisikan secara jelas semua elemen yang akan dipantau. Ini mencakup metrik kinerja aplikasi, seperti waktu respon, throughput, dan tingkat kesalahan (error rate). Selain itu, sumber daya sistem yang digunakan, seperti CPU, memori, dan bandwidth jaringan, juga harus dimasukkan dalam rencana. Rencana ini harus mempertimbangkan semua aspek yang dapat mempengaruhi performa dan ketersediaan aplikasi, serta data terkait yang diperlukan untuk analisis lebih lanjut, seperti log aktivitas pengguna dan data transaksi.

2. Bagaimana melakukan monitoring: Rencana ini mencakup metodologi dan alat yang akan digunakan untuk melakukan monitoring. Ini bisa melibatkan penggunaan software monitoring spesifik, seperti aplikasi pemantauan server, alat analitik untuk aplikasi, atau sistem manajemen log. Selain itu, tim harus menentukan frekuensi pemantauan (misalnya, pemantauan real-time, harian, atau mingguan) dan cara pengumpulan data (apakah secara otomatis atau manual). Di sini juga penting untuk menetapkan prosedur untuk respon terhadap kondisi yang terdeteksi, termasuk rencana untuk eskalasi masalah kepada tim teknis jika diperlukan.
3. Bagaimana hasil dari monitoring dilaporkan dan dapat dimanfaatkan: Setelah data dikumpulkan, rencana monitoring harus mencakup proses pelaporan hasil. Ini bisa melibatkan penyusunan laporan rutin yang memberikan gambaran umum tentang kinerja sistem, analisis tren, dan penilaian risiko. Selain itu, hasil monitoring harus disajikan dengan cara yang mudah dipahami, seperti grafik, dashboard interaktif, atau ringkasan analitik yang mencakup rekomendasi tindakan yang perlu diambil. Hasil monitoring harus dimanfaatkan untuk pengambilan keputusan yang lebih baik terkait pengelolaan sumber daya, perencanaan kapasitas, dan pengembangan fitur baru. Dengan menganalisis hasil monitoring, tim dapat mengidentifikasi area untuk perbaikan, merencanakan upgrade sistem, atau menyesuaikan strategi operasional agar lebih efisien dan responsif terhadap kebutuhan pengguna.

Dengan menyusun rencana monitoring yang komprehensif, organisasi dapat memastikan bahwa pemantauan sistem dilakukan dengan cara yang sistematis, terstruktur, dan memberikan manfaat strategis bagi kinerja aplikasi dan kepuasan pengguna. Elemen-elemen rencana monitoring terdiri dari:

1. Memonitor ambang batas sumber daya: Ini melibatkan penentuan ambang batas nilai untuk setiap sumber daya yang digunakan dalam aplikasi. Ambang batas ini berfungsi sebagai indikator bahwa ada potensi masalah yang perlu ditangani. Misalnya, ambang batas penggunaan sumber daya prosesor adalah 80% saat aplikasi berjalan. Jika penggunaan prosesor mencapai atau melebihi ambang batas ini, tim IT dapat melakukan analisis lebih lanjut untuk mencari penyebabnya dan mengambil tindakan yang diperlukan untuk mencegah dampak negatif pada kinerja aplikasi.
2. Memonitor unjuk kerja: Elemen ini berfokus pada pengukuran dan analisis kinerja aplikasi serta komponen-komponennya. Tim perlu mendefinisikan ukuran unjuk kerja yang relevan, seperti waktu respons aplikasi, throughput, dan latensi. Misalnya, mencatat aktivitas login pemakai dan mencatat pula jumlah kegagalan login pemakai. Data ini sangat penting untuk mengevaluasi efisiensi dan efektivitas aplikasi, serta untuk mengidentifikasi area yang perlu diperbaiki.
3. Menganalisa kecenderungan: Elemen ini melibatkan analisis data yang telah disimpan selama proses monitoring untuk mengidentifikasi pola dan kecenderungan dalam penggunaan aplikasi. Dengan cara ini, tim dapat memprediksi kecenderungan sejumlah pemakai dalam menggunakan bagian-bagian dari aplikasi setiap harinya.

Informasi ini sangat berharga untuk perencanaan kapasitas dan pengembangan fitur baru yang sesuai dengan kebutuhan pengguna.

4. Pendeteksian kegagalan: Ini menjelaskan bagaimana tim dapat memverifikasi bahwa proses pengembangan, operasi, dan perawatan aplikasi sesuai dengan spesifikasi fungsional dan kriteria penerimaan yang telah ditetapkan. Misalnya, jika konsumen yang berbelanja online tidak dapat melihat-lihat item atau daftar item tidak menampilkan produk yang seharusnya, maka kejadian ini akan ditandai sebagai kegagalan. Tim harus memiliki prosedur yang jelas untuk mendokumentasikan dan menangani kegagalan ini agar dapat segera memperbaiki masalah yang muncul.
5. Pendeteksian kesalahan: Elemen ini menjelaskan proses, metode, dan sarana yang akan digunakan untuk mendeteksi dan mendiagnosis kesalahan dalam solusi. Tim harus menetapkan pendekatan yang sistematis untuk mengidentifikasi kesalahan, baik melalui pengujian otomatis maupun manual, serta menggunakan alat bantu yang memadai untuk melakukan diagnosis yang cepat dan akurat. Hal ini sangat penting untuk menjaga kualitas dan keandalan aplikasi.
6. Pencatatan kejadian: Menguraikan proses pencatatan kejadian-kejadian dalam sistem untuk membantu tim dalam melakukan capturing dan reviewing aplikasi secara signifikan. Misalnya, menggunakan SQL Server untuk pencatatan kejadian-kejadian aplikasi, seperti login, transaksi, dan kesalahan yang terjadi. Data ini akan menjadi referensi penting untuk analisis lebih lanjut.
7. Notifikasi: Ini menjelaskan bagaimana operasi yang dilakukan oleh tim akan diinformasikan ketika memonitor dan menelusuri eksepsi yang terdeteksi dalam solusi. Tim harus menggunakan sistem notifikasi yang efektif agar dapat merespons secara cepat terhadap masalah yang muncul.
8. Tools: Menguraikan sarana dan prasarana yang akan digunakan oleh tim untuk mendeteksi, mendiagnosis, dan membetulkan kesalahan serta untuk meningkatkan unjuk kerja aplikasi. Pilihan tools yang tepat, seperti software monitoring, sistem logging, dan alat analisis performa, dapat sangat membantu dalam memastikan aplikasi tetap berfungsi dengan baik dan memenuhi ekspektasi pengguna.

Dengan mencakup elemen-elemen ini dalam rencana monitoring, organisasi dapat memastikan bahwa aplikasi tetap dalam kondisi optimal, serta siap untuk menghadapi tantangan yang mungkin muncul selama operasionalnya.

Perencanaan Fitur Migrasi Data

Perencanaan Fitur Migrasi data adalah proses memindahkan atau merubah data dari format yang digunakan sebelumnya ke format data yang baru. Proses ini sangat penting ketika sebuah aplikasi baru dibangun, terutama ketika aplikasi tersebut menggantikan atau memperbarui aplikasi yang sudah ada. Oleh karena itu, identifikasi data yang digunakan pada aplikasi sebelumnya menjadi langkah awal yang krusial untuk memastikan bahwa migrasi data dapat dilakukan dengan baik dan efisien. Dalam perencanaan migrasi data, beberapa langkah penting perlu dipertimbangkan:

1. Identifikasi Sumber Data: Tim harus mengidentifikasi semua sumber data yang ada di aplikasi lama, termasuk database, file, dan sistem penyimpanan lain yang relevan. Pemetaan data ini akan membantu dalam memahami struktur dan hubungan antar data yang ada.
2. Analisis Kualitas Data: Sebelum migrasi dilakukan, penting untuk menganalisis kualitas data yang ada. Ini termasuk memeriksa data duplikat, kesalahan, dan inkonsistensi. Memastikan kualitas data yang baik akan mengurangi masalah yang mungkin timbul setelah migrasi dan meningkatkan kinerja aplikasi baru.
3. Perencanaan Transformasi Data: Proses migrasi tidak hanya melibatkan pemindahan data, tetapi juga mungkin memerlukan transformasi data agar sesuai dengan struktur dan format baru dalam aplikasi yang sedang dikembangkan. Tim harus mendefinisikan aturan transformasi yang diperlukan, termasuk pengubahan tipe data, normalisasi, dan penggabungan informasi dari beberapa sumber.
4. Pembuatan Rencana Migrasi: Setelah mengidentifikasi dan menganalisis data, langkah berikutnya adalah membuat rencana migrasi yang jelas. Rencana ini harus mencakup langkah-langkah untuk pemindahan data, jadwal, dan tanggung jawab setiap anggota tim. Rencana ini juga harus mempertimbangkan waktu henti yang mungkin diperlukan selama proses migrasi untuk meminimalkan gangguan pada pengguna.
5. Uji Coba Migrasi: Sebelum melakukan migrasi secara penuh, sangat penting untuk melakukan uji coba migrasi. Hal ini dilakukan untuk memastikan bahwa proses migrasi berjalan lancar dan data yang dipindahkan dapat berfungsi dengan baik di sistem baru. Uji coba ini juga memberikan kesempatan untuk mengidentifikasi dan memperbaiki potensi masalah sebelum migrasi final.
6. Pelaksanaan Migrasi: Setelah semua persiapan dilakukan dan uji coba berhasil, proses migrasi dapat dilaksanakan. Selama migrasi, tim harus memantau proses dengan cermat untuk memastikan bahwa semua data dipindahkan dengan benar dan tidak ada kehilangan data.
7. Verifikasi dan Validasi: Setelah migrasi selesai, penting untuk melakukan verifikasi dan validasi untuk memastikan bahwa semua data telah dipindahkan dengan benar dan aplikasi baru berfungsi sebagaimana mestinya. Ini termasuk pengujian fungsi aplikasi, integritas data, dan kinerja sistem.
8. Dokumentasi dan Pelatihan: Akhirnya, mendokumentasikan seluruh proses migrasi dan hasil yang dicapai sangatlah penting. Selain itu, tim juga harus menyediakan pelatihan untuk pengguna akhir agar mereka dapat beradaptasi dengan aplikasi baru dengan lebih mudah.

Rencana migrasi data adalah proses terencana dan terstruktur yang digunakan untuk memindahkan data dari satu sistem atau lingkungan ke sistem atau lingkungan lainnya. Rencana ini sangat penting untuk memastikan bahwa data yang dipindahkan tetap utuh, akurat, dan dapat diakses setelah migrasi. Berikut adalah penjelasan rinci mengenai komponen utama dari rencana migrasi data:

1. Strategi Migrasi

Strategi migrasi adalah rencana besar yang menguraikan pendekatan yang akan diambilselama proses migrasi data. Ini meliputi:

1. Penilaian Data: Mengidentifikasi data mana yang perlu dipindahkan, serta kualitas dan format data yang ada.
2. Pendekatan Migrasi: Memilih antara beberapa pendekatan, seperti:
3. Big Bang Migration: Memindahkan semua data dalam satu waktu tertentu,
4. Phased Migration: Memindahkan data secara bertahap dalam beberapa fase untuk mengurangi risiko dan gangguan.
5. Rencana Waktu: Menyusun jadwal migrasi yang jelas, termasuk tanggal dan waktu untuk setiap fase migrasi.

2. Peralatan

Peralatan migrasi adalah alat dan teknologi yang digunakan untuk mendukung dan memfasilitasi proses migrasi. Ini mencakup:

1. Software Migrasi: Alat perangkat lunak khusus yang membantu dalam memindahkan data, seperti ETL (*Extract, Transform, Load*) tools, yang dapat mengekstrak data dari sumber, mengubah formatnya, dan memuatnya ke dalam sistem tujuan.
2. Database Management Systems: Sistem manajemen basis data yang digunakan untuk menyimpan dan mengelola data dalam kedua lingkungan, sumber dan tujuan.
3. Monitoring Tools: Alat untuk memantau dan melacak kemajuan migrasi serta untuk mendeteksi dan menyelesaikan masalah yang mungkin muncul.

3. Petunjuk Migrasi

Petunjuk migrasi berisi langkah-langkah detail tentang bagaimana melakukan migrasi data. Ini mencakup:

1. Persiapan Data: Langkah-langkah untuk membersihkan, memformat, dan memvalidasi data sebelum migrasi.
2. Ekstraksi Data: Proses untuk mengekstrak data dari sumber.
3. Transformasi Data: Jika diperlukan, langkah-langkah untuk mengubah format atau struktur data agar sesuai dengan sistem tujuan.
4. Pemindahan Data: Menggunakan peralatan migrasi untuk memindahkan data ke sistem tujuan.
5. Verifikasi Data: Setelah pemindahan, melakukan pemeriksaan untuk memastikan data telah dipindahkan dengan benar dan lengkap.

4. Proses Migrasi

Proses migrasi adalah langkah-langkah praktis yang dilakukan selama migrasi data. Ini mencakup:

1. Pelaksanaan: Melaksanakan rencana migrasi sesuai dengan petunjuk yang telah ditetapkan.
2. Pengawasan: Memantau proses migrasi untuk memastikan bahwa tidak ada masalah atau kesalahan yang terjadi selama pemindahan data.

3. Dokumentasi: Mencatat setiap langkah, termasuk masalah yang muncul dan bagaimana cara mengatasinya untuk referensi di masa mendatang.

5. Pengujian Lingkungan Migrasi

Lingkungan pengujian migrasi adalah replikasi dari lingkungan produksi yang sesungguhnya, yang digunakan untuk:

1. Pengujian Migrasi: Melakukan migrasi data di lingkungan pengujian terlebih dahulu untuk memastikan semua langkah berfungsi dengan baik sebelum dilakukan di lingkungan produksi.
2. Validasi Hasil: Memeriksa apakah data telah berhasil dipindahkan dan berfungsi dengan benar di lingkungan pengujian.
3. Identifikasi Masalah: Mendeteksi dan memperbaiki masalah yang muncul sebelum migrasi dilakukan secara nyata.

6. Pembatalan Konfigurasi ke Keadaan Sebelumnya

Jika terjadi kesalahan atau masalah selama proses migrasi, penting untuk memiliki rencana pemulihan:

1. Rollback Plan: Menyusun rencana untuk mengembalikan sistem ke keadaan sebelumnya jika migrasi tidak berhasil. Ini termasuk mencadangkan data sebelum migrasi dilakukan sehingga dapat dikembalikan jika diperlukan.
2. Dokumentasi Kesalahan: Mencatat kesalahan yang terjadi dan langkah-langkah yang diambil untuk memperbaikinya, sehingga dapat digunakan sebagai panduan untuk perbaikan di masa depan.

Membuat Spesifikasi Lisensi

Spesifikasi lisensi sangat penting dalam pengembangan solusi aplikasi, mencakup perangkat lunak dan perangkat keras yang dibutuhkan agar operasional aplikasi sesuai dengan ketentuan hukum dan operasional. Langkah pertama adalah mengidentifikasi kebutuhan perangkat lunak seperti sistem operasi, database, alat pengembangan, serta perangkat keras seperti server atau perangkat IoT. Kemudian, perlu dipahami jenis lisensi perangkat lunak yang akan digunakan, misalnya lisensi open source (GPL, MIT, Apache), lisensi komersial, atau berbasis langganan, yang masing-masing memiliki batasan dan ketentuan khusus. Penting pula memastikan bahwa penggunaan perangkat sesuai dengan ketentuan lisensi, baik untuk perangkat lunak maupun perangkat keras, yang mungkin memiliki batasan seperti kepatuhan pada jumlah pengguna atau jenis penggunaan (komersial atau non-komersial). Selain itu, dokumentasi lengkap untuk setiap lisensi perangkat yang digunakan, termasuk informasi ketentuan utama, tanggal berlaku, dan kebutuhan pembaruan, membantu memastikan kepatuhan hukum. Proses pemantauan penggunaan lisensi yang ketat akan membantu memastikan bahwa perangkat lunak dan perangkat keras digunakan sesuai dengan persyaratan, menjaga aplikasi dari potensi masalah hukum, serta melindungi hak cipta setiap komponen yang digunakan.

Menetapkan spesifikasi perangkat lunak dan perangkat keras di awal proses pengembangan aplikasi memiliki berbagai manfaat strategis dan operasional yang signifikan. Pertama, spesifikasi ini membantu **memastikan proses persetujuan di setiap tahap**

perancangan dengan menetapkan standar yang harus dipenuhi oleh setiap komponen perangkat. Dengan demikian, tim pengembang dapat menilai apakah perangkat lunak dan perangkat keras yang dipilih sesuai dengan kebutuhan teknis, keamanan, dan kepatuhan yang telah ditentukan. Proses ini juga mengurangi potensi konflik atau masalah yang mungkin timbul saat integrasi perangkat baru ke dalam sistem, sehingga seluruh tim bekerja secara harmonis dan terarah pada spesifikasi yang sudah disetujui bersama.

Selain itu, menetapkan spesifikasi sejak awal memberikan **waktu yang memadai bagi vendor untuk mensuplai perangkat keras dan perangkat lunak** sesuai kebutuhan proyek. Ini menghindari kemungkinan penundaan yang sering kali muncul ketika komponen yang diperlukan tidak tersedia tepat waktu. Dengan spesifikasi yang jelas, vendor dapat memahami jadwal proyek dan menyiapkan pasokan sesuai kebutuhan sehingga mereka dapat memenuhi tenggat waktu yang telah ditentukan tanpa mengganggu jalannya pengembangan atau penjadwalan proyek secara keseluruhan. Dengan demikian, penetapan spesifikasi perangkat lunak dan perangkat keras membantu meminimalkan risiko keterlambatan, menjaga konsistensi dalam desain dan standar operasional, serta memastikan keterlibatan vendor dalam menyediakan solusi tepat waktu.

Aspek lisensi sangat penting dalam spesifikasi pembelian perangkat lunak dan perangkat keras karena menentukan hak dan batasan penggunaan, instalasi, dan distribusi. Lisensi perangkat lunak, seperti open source, komersial, atau berbasis langganan, memiliki syarat berbeda yang memengaruhi fleksibilitas, biaya, dan keamanan operasional. Begitu juga dengan perangkat keras yang sering kali memiliki perangkat lunak bawaan dan lisensi tertentu, terutama pada perangkat IoT dan jaringan. Memahami lisensi dengan baik membantu organisasi tetap patuh secara hukum, menghindari pelanggaran, dan memastikan kelancaran operasional serta keberlanjutan dukungan di masa depan.

Spesifikasi lisensi sangat diperlukan pada tahap pengembangan dan penyebaran solusi aplikasi karena menentukan hak, batasan, dan kewajiban yang berkaitan dengan penggunaan perangkat lunak dan perangkat keras yang digunakan selama siklus hidup aplikasi. Pada tahap pengembangan, tim biasanya mengandalkan berbagai alat dan pustaka perangkat lunak yang memiliki lisensi berbeda-beda. Tanpa lisensi yang sesuai, penggunaan produk tersebut bisa menimbulkan masalah hukum, mengganggu alur pengembangan, atau bahkan membatasi kemampuan pengembang untuk mengadaptasi dan memodifikasi perangkat lunak sesuai kebutuhan proyek.

Lisensi perangkat lunak mengatur hak akses dan modifikasi kode serta memastikan bahwa penggunaan alat dan pustaka yang dipakai tidak melanggar aturan hukum atau ketentuan dari pemilik hak cipta. Misalnya, lisensi open source seperti GNU *General Public License* (GPL) atau MIT License memungkinkan modifikasi kode sumber, namun beberapa lisensi mungkin memiliki syarat tertentu yang harus dipatuhi. Sementara itu, lisensi komersial dapat memberikan fitur tambahan atau dukungan teknis, namun sering kali membatasi kemampuan modifikasi dan penggunaan ulang di luar ketentuan yang disepakati. Dengan memastikan lisensi yang sah di tahap pengembangan, tim dapat bekerja dengan fleksibilitas

lebih tinggi, tetap patuh pada aturan, dan bebas dari hambatan hukum atau pembatasan lisensi yang bisa merugikan proyek di masa depan.

Pada tahap penyebaran, spesifikasi lisensi juga diperlukan untuk memastikan bahwa seluruh perangkat lunak dan perangkat keras yang digunakan dalam lingkungan produksi memiliki izin yang sesuai. Sebagian besar perangkat lunak komersial memiliki ketentuan yang berbeda untuk penggunaan pengembangan dan produksi, sehingga lisensi yang digunakan dalam lingkungan produksi sering kali memerlukan pembelian atau kontrak tambahan. Hal ini sangat penting bagi aplikasi yang berskala besar atau yang digunakan untuk tujuan komersial, karena pelanggaran lisensi di tahap ini bisa menyebabkan sanksi hukum atau biaya tak terduga. Misalnya, lisensi pengembangan yang mungkin gratis atau berbiaya rendah untuk uji coba, mungkin memerlukan pembelian lisensi penuh saat aplikasi diluncurkan untuk publik atau pengguna komersial.

Lebih jauh lagi, spesifikasi lisensi yang lengkap dan terstruktur memungkinkan organisasi untuk mengelola hak-hak penggunaan setiap komponen dalam aplikasi dengan baik, sehingga memastikan bahwa aplikasi dapat berfungsi secara penuh sesuai dengan lisensi yang berlaku. Ini juga membantu dalam mendokumentasikan seluruh ketentuan lisensi untuk tujuan pemantauan dan kepatuhan jangka panjang, menjaga agar penggunaan perangkat lunak tetap legal dan sesuai dengan ketentuan operasional

Selama tahap pengembangan solusi aplikasi, tim pengembang biasanya akan mengandalkan berbagai teknologi dan produk perangkat lunak yang telah dipilih berdasarkan kebutuhan fungsional, teknis, dan anggaran proyek. Penting untuk memastikan bahwa setiap perangkat lunak dan alat yang digunakan telah memiliki lisensi yang sah agar tidak melanggar hak cipta dan peraturan hukum. Proses ini mencakup beberapa aspek penting yang perlu diperhatikan.

1. Identifikasi Kebutuhan Perangkat Lunak:
- Tim harus terlebih dahulu mengidentifikasi perangkat lunak dan teknologi yang diperlukan untuk memenuhi tujuan pengembangan. Ini termasuk memilih *Integrated Development Environments* (IDEs), framework, pustaka pemrograman, sistem manajemen basis data, serta alat pengujian dan manajemen proyek.

2. Memahami Jenis Lisensi:
- Setiap produk perangkat lunak memiliki jenis lisensi yang berbeda, yang mengatur bagaimana perangkat lunak tersebut dapat digunakan. Ini dapat mencakup lisensi open source, lisensi komersial, dan lisensi berbasis langganan.
- **Lisensi Open Source:** Lisensi ini memungkinkan pengguna untuk menggunakan, memodifikasi, dan mendistribusikan perangkat lunak secara bebas. Contoh lisensi open source adalah GPL, MIT, dan Apache. Namun, pengguna harus memahami syarat dan ketentuan yang berlaku, seperti kewajiban untuk menyertakan kode sumber atau mencantumkan atribusi.
- **Lisensi Komersial:** Lisensi ini biasanya memerlukan pembayaran dan memberikan hak terbatas kepada pengguna. Pengguna sering kali tidak diperbolehkan untuk memodifikasi atau mendistribusikan ulang perangkat lunak tanpa izin.

- **Lisensi Berbasis Langganan:** Lisensi ini memungkinkan akses ke perangkat lunak untuk jangka waktu tertentu dengan biaya bulanan atau tahunan, sering kali mencakup pembaruan dan dukungan teknis.

3. Memastikan Kepatuhan Lisensi:

- Setelah mengidentifikasi dan memilih produk perangkat lunak, tim harus memverifikasi bahwa mereka memiliki lisensi yang sesuai untuk setiap produk yang akan digunakan dalam pengembangan. Ini termasuk melakukan audit terhadap perangkat lunak yang ada dan memeriksa bahwa semua lisensi yang diperlukan telah diperoleh.

4. Mengelola Risiko Hukum:

- Menggunakan perangkat lunak tanpa lisensi yang tepat dapat mengakibatkan risiko hukum yang signifikan, termasuk denda, litigasi, dan kerusakan reputasi. Oleh karena itu, kepatuhan lisensi yang baik sangat penting untuk menjaga integritas proyek dan organisasi.

5. Pendidikan dan Pelatihan:

- Untuk memastikan bahwa seluruh anggota tim memahami pentingnya kepatuhan lisensi, pelatihan dan pendidikan tentang lisensi perangkat lunak dan implikasinya perlu diberikan. Ini membantu meningkatkan kesadaran dan memastikan bahwa semua anggota tim bertanggung jawab dalam menggunakan perangkat lunak sesuai dengan ketentuan yang berlaku.

6. Proses Pembaruan dan Evaluasi:

- Tim juga harus memiliki proses untuk mengevaluasi dan memperbarui lisensi secara berkala, terutama ketika perangkat lunak baru ditambahkan atau ketika ada perubahan dalam lisensi yang ada. Memastikan lisensi selalu diperbarui dan relevan membantu menghindari masalah di masa depan.

Merencanakan Tahapan Kedepan (Pengembangan Dan Penyebaran)

Merencanakan tahapan kedepan dalam pengembangan dan penyebaran solusi aplikasi merupakan langkah krusial yang harus dilakukan oleh tim pengembang sebelum mereka mulai bekerja. Proses ini tidak hanya mencakup persiapan teknis, tetapi juga perencanaan strategis yang mencakup pengelolaan risiko, pengendalian biaya, dan evaluasi kinerja. Berikut adalah beberapa poin penting dalam merencanakan tahapan kedepan:

1. Verifikasi Lingkungan Pengembangan dan Pengujian:

- ✓ Sebelum memulai pengembangan, penting untuk memastikan bahwa lingkungan pengembangan dan pengujian telah siap dan berfungsi dengan baik. Lingkungan ini harus dapat mereplikasi kondisi nyata yang akan ada di lingkungan produksi untuk memfasilitasi pengujian yang efektif dan realistis.
- ✓ Tim harus memeriksa semua komponen teknis, termasuk server, database, perangkat lunak yang diperlukan, serta konfigurasi jaringan. Ini termasuk memastikan bahwa perangkat lunak telah terinstal dengan benar dan semua dependensi telah dipenuhi.

2. Desain Lingkungan yang Sesuai:
   - Lingkungan pengembangan harus dirancang untuk mencerminkan lingkungan produksi dengan seakurat mungkin. Hal ini memungkinkan pengujian yang lebih baik dan lebih relevan, serta membantu mengidentifikasi potensi masalah sebelum solusi diluncurkan ke pengguna akhir.
   - Namun, tim juga harus memperhatikan faktor biaya. Membangun lingkungan pengembangan yang terlalu mirip dengan lingkungan produksi dapat mengakibatkan pengeluaran yang tidak perlu. Oleh karena itu, perlu ada keseimbangan antara kesesuaian lingkungan dan biaya yang dikeluarkan.
3. Perencanaan untuk Pengujian:
   - Pengujian merupakan bagian penting dari pengembangan perangkat lunak, dan perencanaan harus mencakup strategi untuk berbagai jenis pengujian, seperti pengujian unit, integrasi, sistem, dan penerimaan. Setiap jenis pengujian harus dilakukan dalam lingkungan yang sesuai untuk memastikan hasil yang akurat.
   - Tim juga harus mempertimbangkan pembuatan skrip otomatisasi pengujian untuk menghemat waktu dan sumber daya, serta meningkatkan konsistensi dan keandalan hasil pengujian.
4. Pengelolaan Risiko:
   - Dalam merencanakan tahapan pengembangan, penting untuk mengidentifikasi dan menganalisis potensi risiko yang mungkin terjadi. Ini termasuk risiko teknis, risiko anggaran, dan risiko waktu.
   - Tim harus mengembangkan strategi mitigasi untuk mengatasi risiko tersebut, termasuk membuat rencana cadangan yang jelas jika terjadi masalah selama pengembangan atau penyebaran.
5. Penerapan Metodologi Pengembangan:
   - Memilih metodologi pengembangan yang tepat, seperti Agile, Waterfall, atau DevOps, sangat penting dalam merencanakan tahapan kedepan. Metodologi ini akan memandu bagaimana tim bekerja, berkomunikasi, dan beradaptasi terhadap perubahan yang mungkin terjadi selama proses pengembangan.
   - Dalam metodologi Agile, misalnya, tim harus siap untuk iterasi yang cepat dan umpan balik yang berkelanjutan, sedangkan metodologi Waterfall mungkin memerlukan perencanaan yang lebih detail di awal.
6. Pelibatan Pemangku Kepentingan:
   - Penting untuk melibatkan pemangku kepentingan dari awal perencanaan hingga penyebaran. Hal ini membantu memastikan bahwa semua kebutuhan dan harapan terkait dengan solusi aplikasi dipahami dan diakomodasi selama proses pengembangan.
   - Komunikasi yang efektif dengan pemangku kepentingan dapat membantu dalam menetapkan prioritas dan menyelesaikan masalah yang mungkin muncul dengan cepat.

7. Dokumentasi:
- Dokumentasi adalah aspek penting dalam perencanaan. Setiap keputusan yang diambil, proses yang dilalui, dan hasil yang dicapai harus didokumentasikan dengan baik. Ini tidak hanya berguna untuk referensi di masa depan, tetapi juga membantu tim dalam pembelajaran dan peningkatan berkelanjutan.

8. Pengendalian Biaya:
- Mengelola biaya selama tahap pengembangan dan penyebaran sangat penting. Tim harus membuat anggaran yang realistis dan mengawasi pengeluaran untuk memastikan bahwa proyek tetap dalam batas anggaran yang telah ditetapkan.
- Ini juga melibatkan pengawasan terhadap biaya sumber daya manusia, perangkat lunak, perangkat keras, dan biaya operasional lainnya.

Elemen Proses Pengembangan :

1.Tujuan Pengembangan:
- ✓ **Definisi:** Tujuan pengembangan mencakup apa yang ingin dicapai oleh tim pengembang melalui proyek ini. Ini bisa meliputi kebutuhan bisnis, fungsionalitas yang harus ada dalam perangkat lunak, dan hasil akhir yang diharapkan.
- ✓ **Konteks:** Menentukan tujuan dengan jelas membantu tim dalam memfokuskan usaha dan sumber daya mereka. Tujuan yang jelas juga memfasilitasi komunikasi dengan pemangku kepentingan dan memastikan bahwa semua pihak memiliki pemahaman yang sama tentang apa yang akan dicapai.

2. Strategi Penyampaian Keseluruhan Solusi:
- ✓ **Definisi:** Ini merujuk pada rencana umum yang digunakan untuk mengembangkan dan menyampaikan solusi perangkat lunak. Strategi ini mencakup pemilihan metode pengembangan (Agile, Waterfall, dll.) dan cara solusi akan diimplementasikan.
- ✓ **Konteks:** Strategi yang dipilih akan mempengaruhi bagaimana tim berkolaborasi, bagaimana umpan balik dikumpulkan, dan bagaimana perubahan dapat diimplementasikan selama proses pengembangan.

3. Pendekatan Tradeoff:
- ✓ **Definisi:** Dalam pengembangan perangkat lunak, sering kali ada trade-off antara berbagai faktor seperti biaya, waktu, kualitas, dan ruang lingkup. Pendekatan trade-off mengharuskan tim untuk menganalisis dan membuat keputusan yang seimbang.
- ✓ **Konteks:** Memahami trade-off yang ada memungkinkan tim untuk memprioritaskan fitur yang paling penting dan mengelola ekspektasi pemangku kepentingan. Ini juga membantu dalam menyesuaikan rencana jika diperlukan.

4. Kunci Utama Desain:
- ✓ **Definisi:** Ini mencakup prinsip dan keputusan desain yang menjadi dasar pengembangan solusi. Hal ini termasuk arsitektur perangkat lunak, antarmuka pengguna, dan pengalaman pengguna (UX).
- ✓ **Konteks:** Kunci utama desain harus sejalan dengan tujuan proyek dan memenuhi kebutuhan pengguna. Desain yang baik dapat meningkatkan keterlibatan pengguna dan memfasilitasi penggunaan aplikasi yang lebih efisien.

5. Pengembangan dan Lingkungannya:
- ✓ **Definisi:** Merujuk pada pengaturan teknis di mana pengembangan perangkat lunak dilakukan, termasuk perangkat keras, perangkat lunak, dan infrastruktur jaringan yang diperlukan.
- ✓ **Konteks:** Lingkungan pengembangan yang tepat sangat penting untuk efektivitas proses pengembangan. Lingkungan ini harus dapat mensimulasikan kondisi nyata yang akan ada dalam penggunaan akhir.

6. Petunjuk dan Standarisasi:
- ✓ **Definisi:** Ini mencakup pedoman dan praktik terbaik yang harus diikuti oleh tim selama proses pengembangan, termasuk standar kode, dokumentasi, dan proses pengujian.
- ✓ **Konteks:** Petunjuk dan standar yang jelas membantu menjaga konsistensi dalam pengembangan, memudahkan pemeliharaan dan kolaborasi antar anggota tim.

7. Kontrol Kode dan Pembuatan Versi:
- ✓ **Definisi:** Proses yang digunakan untuk mengelola perubahan dalam kode sumber perangkat lunak, termasuk penggunaan sistem kontrol versi seperti Git untuk melacak perubahan dan kolaborasi.
- ✓ **Konteks:** Kontrol kode yang efektif memungkinkan pengembang untuk bekerja secara paralel, mengelola konflik, dan melacak riwayat perubahan yang dilakukan pada proyek. Ini juga mendukung rollback jika terjadi kesalahan.

8. Proses Pembangunan Solusi:
- ✓ **Definisi:** Ini mencakup langkah-langkah spesifik yang diambil untuk membangun dan menyusun solusi perangkat lunak, dari pengkodean hingga pengujian dan implementasi.
- ✓ **Konteks:** Proses pembangunan harus terencana dengan baik dan diikuti dengan disiplin untuk menghasilkan produk yang berkualitas dan siap digunakan.

9. Komponen-Komponen Solusi:
- ✓ **Definisi:** Komponen solusi adalah bagian-bagian individu dari perangkat lunak yang bekerja sama untuk memberikan fungsi keseluruhan. Ini termasuk modul, pustaka, dan layanan.
- ✓ **Konteks:** Mengidentifikasi dan merancang komponen-komponen ini dengan baik dapat mempermudah pengembangan, pemeliharaan, dan pengujian.

10. Manajemen Pengembangan dan Konfigurasinya:
- ✓ **Definisi:** Proses mengelola dan mengontrol semua aspek pengembangan, termasuk pengaturan sumber daya, jadwal, dan pengelolaan konfigurasi perangkat lunak.
- ✓ **Konteks:** Manajemen yang baik memastikan bahwa proyek tetap pada jalurnya, mengurangi risiko keterlambatan, dan meningkatkan efisiensi kerja tim.

11. Design Pattern (Desain Class Spesifik sebagai Template):

- ✓ **Definisi:** Pola desain adalah solusi yang teruji untuk masalah umum yang dihadapi dalam pengembangan perangkat lunak. Pola ini dapat berupa template untuk arsitektur atau struktur kelas.
- ✓ **Konteks:** Menggunakan pola desain yang tepat dapat mempercepat proses pengembangan, meningkatkan kualitas kode, dan mempermudah pemeliharaan serta pengembangan lebih lanjut.

12. Pelatihan dan Dukungan Tim Pengembang:

- ✓ **Definisi:** Pelatihan dan dukungan diperlukan untuk memastikan bahwa anggota tim memiliki keterampilan dan pengetahuan yang diperlukan untuk menyelesaikan proyek.
- ✓ **Konteks:** Memberikan pelatihan yang sesuai dapat meningkatkan produktivitas tim, mengurangi kesalahan, dan meningkatkan kepuasan kerja. Dukungan yang baik juga penting untuk menyelesaikan masalah teknis atau tantangan yang dihadapi selama pengembangan.

Spesifikasi Teknis

Merupakan dokumen penting dalam pengembangan perangkat lunak yang berfungsi sebagai panduan dan referensi bagi tim pengembang. Dokumen ini biasanya mencakup rincian tentang desain fisikal sistem, komponen yang diperlukan, dan bagaimana semua elemen dalam solusi perangkat lunak akan berinteraksi satu sama lain. Berikut adalah penjelasan rinci mengenai elemen-elemen dalam spesifikasi teknis dan peranannya dalam proses pengembangan:

1. ***Spesifikasi Kelas***

- **Definisi:** Spesifikasi kelas mendefinisikan struktur dan perilaku dari kelas-kelas yang ada dalam perangkat lunak. Ini mencakup atribut, metode, dan relasi antara kelas.
- **Peran:** Dengan mendokumentasikan spesifikasi kelas, tim dapat memastikan bahwa setiap kelas dirancang dengan baik, memudahkan pengembang dalam mengimplementasikan fungsionalitas dan memelihara kode di masa mendatang. Spesifikasi ini juga membantu dalam memahami hierarki dan interaksi antar kelas.

2. ***Model Komponen***

- **Definisi:** Model komponen menjelaskan komponen-komponen perangkat lunak yang ada, serta hubungan dan interaksi antar komponen tersebut. Ini sering kali digambarkan dalam bentuk diagram yang menunjukkan bagaimana komponen saling terhubung.
- **Peran:** Model ini membantu tim dalam memahami arsitektur sistem secara keseluruhan. Dengan model yang jelas, pengembang dapat melihat bagaimana setiap komponen berkontribusi pada fungsionalitas aplikasi dan mengidentifikasi area yang mungkin perlu perhatian khusus.

3. ***Metrics***

- **Definisi:** Metrics adalah pengukuran yang digunakan untuk mengevaluasi berbagai aspek perangkat lunak, seperti kinerja, keandalan, dan kompleksitas. Ini dapat mencakup jumlah baris kode, waktu respons, dan tingkat kesalahan.

- **Peran:** Menetapkan metrics memungkinkan tim untuk melakukan penilaian objektif terhadap kualitas dan efisiensi perangkat lunak. Metrics dapat digunakan untuk mengidentifikasi area yang perlu ditingkatkan dan untuk melacak kemajuan selama pengembangan.

4. ***Jaringan dan Komponen Teknologi***

- **Definisi:** Bagian ini menjelaskan infrastruktur jaringan yang diperlukan untuk menjalankan perangkat lunak, termasuk server, basis data, dan teknologi komunikasi yang digunakan.
- **Peran:** Memahami komponen jaringan dan teknologi yang terlibat sangat penting untuk memastikan bahwa solusi dapat diimplementasikan dalam lingkungan yang tepat. Ini juga mencakup pertimbangan terkait keamanan, performa, dan skalabilitas.

Penggunaan Spesifikasi Teknis dalam Tahap Pengembangan

Selama tahap pengembangan, spesifikasi teknis berfungsi sebagai panduan utama bagi tim. Berikut adalah beberapa cara spesifikasi teknis digunakan dalam praktik:

- **Definisi Pekerjaan dan Jangkauan:** Spesifikasi teknis membantu tim dalam menetapkan ruang lingkup proyek dan membagi tugas menjadi bagian-bagian yang lebih kecil. Dengan mendefinisikan apa yang perlu dilakukan dan bagaimana melakukannya, spesifikasi ini memberikan kejelasan kepada pengembang tentang tanggung jawab mereka.
- **Definisi Antarmuka:** Spesifikasi ini mencakup deskripsi rinci tentang antarmuka yang harus diimplementasikan. Hal ini meliputi bagaimana komponen berinteraksi dan pertukaran data, memastikan bahwa integrasi antara berbagai bagian sistem berjalan dengan lancar.
- **Entri Registri dan Instalasi:** Dokumen ini juga mencakup detail teknis mengenai entri registri yang diperlukan, ukuran byte untuk instalasi, dan pengaturan lainnya yang diperlukan untuk memastikan perangkat lunak dapat diinstal dan dijalankan tanpa masalah.
- **Dynamic Link Library (DLL) dan Nama Assembly:** Dalam konteks pengembangan perangkat lunak, spesifikasi teknis mencakup informasi tentang DLL yang digunakan dalam aplikasi. Ini termasuk nama assembly, strong name, dan kunci yang diperlukan untuk pengelolaan komponen dan dependensi.
- **Elemen Distribusi:** Semua elemen yang berpengaruh pada tahap penyebaran dan distribusi perangkat lunak juga diuraikan dalam spesifikasi teknis. Ini mencakup persyaratan sistem, proses instalasi, dan prosedur pengujian yang harus diikuti sebelum solusi diluncurkan ke lingkungan produksi.

Dokumen spesifikasi teknis adalah panduan rinci yang merangkum berbagai elemen penting dalam pengembangan solusi aplikasi. Dokumen ini berfungsi sebagai referensi bagi tim pengembang, pemangku kepentingan, dan anggota tim lainnya untuk memastikan bahwa semua aspek teknis dari proyek dipahami dan diimplementasikan dengan benar.

Berikut adalah penjelasan rinci tentang elemen-elemen yang termasuk dalam dokumen spesifikasi teknis:

1. Architecture Overview
   - **Definisi:** Elemen ini memberikan gambaran umum tentang arsitektur sistem yang akan diterapkan. Ini mencakup komponen utama, interaksi antar komponen, dan bagaimana arsitektur mendukung kebutuhan fungsional dan non-fungsional aplikasi.
   - **Peran:** Memahami arsitektur membantu semua pihak dalam proyek untuk memiliki visi yang sama mengenai bagaimana sistem akan dibangun. Arsitektur yang baik akan memfasilitasi skalabilitas, kinerja, dan keamanan sistem.
2. Object Model
   - **Definisi:** Model objek menjelaskan struktur objek dalam solusi, termasuk atribut dan metode yang terkait dengan masing-masing objek. Ini juga mencakup hubungan antar objek, seperti pewarisan dan asosiasi.
   - **Peran:** Dengan mendeskripsikan model objek, pengembang dapat memahami bagaimana data akan diorganisasikan dan dikelola. Ini juga memudahkan implementasi logika bisnis dan interaksi antar komponen dalam aplikasi.
3. Interface
   - **Definisi:** Elemen ini mencakup kode dan detail metode pada setiap antarmuka (interface) yang digunakan dalam solusi. Ini meliputi parameter, tipe data, dan pengembalian dari setiap metode.
   - **Peran:** Spesifikasi antarmuka yang jelas memungkinkan pengembang untuk mengetahui cara berinteraksi dengan komponen lain dalam sistem. Ini juga memfasilitasi integrasi dengan sistem eksternal dan memastikan konsistensi dalam penggunaan metode.
4. Error Logging
   - **Definisi:** Ini menjelaskan berbagai jenis kesalahan yang akan ditangani oleh solusi dan cara mencatat kesalahan tersebut. Ini mencakup pengelolaan kesalahan yang terjadi selama eksekusi dan proses penyimpanan informasi kesalahan.
   - **Peran:** Dengan adanya mekanisme pencatatan kesalahan yang baik, tim dapat dengan cepat mengidentifikasi dan memperbaiki masalah yang muncul. Ini juga memberikan data yang berguna untuk analisis performa dan perbaikan di masa depan.
5. Configuration
   - **Definisi:** Elemen ini menjelaskan bagaimana solusi aplikasi akan didaftarkan (register) pada komputer tujuan, termasuk pengaturan konfigurasi yang diperlukan untuk menjalankan aplikasi dengan benar.
   - **Peran:** Memastikan bahwa konfigurasi aplikasi dijelaskan dengan jelas membantu dalam proses instalasi dan penyebaran. Ini juga penting untuk memastikan bahwa aplikasi berfungsi sesuai harapan di lingkungan yang ditargetkan.
6. Supporting Documentation

- **Definisi:** Ini adalah daftar dokumen yang menguraikan solusi secara lebih mendalam, termasuk spesifikasi fungsional, panduan pengguna, dan lokasi dari dokumen-dokumen tersebut.
- **Peran:** Menyediakan akses ke dokumentasi yang mendukung memungkinkan anggota tim untuk memahami lebih baik fungsionalitas aplikasi dan mempercepat proses pelatihan serta pemeliharaan.

7. Issues

- **Definisi:** Elemen ini mencakup isu-isu yang diketahui mengenai solusi, termasuk masalah yang pernah dihadapi selama pengembangan dan solusi yang diterapkan. Ini biasanya mencakup tanggal untuk setiap spesifikasi pada sesi.
- **Peran:** Dengan mencatat isu-isu yang diketahui, tim dapat meminimalkan risiko di masa depan dan meningkatkan transparansi dalam pengembangan. Ini juga memudahkan manajemen proyek untuk merespons masalah yang muncul secara proaktif.

8. Code Flow

- **Definisi:** Ini mendeskripsikan bagaimana aliran eksekusi setiap metode dalam solusi. Ini mencakup langkah-langkah yang diambil oleh aplikasi dalam menjalankan fungsi tertentu.
- **Peran:** Memahami aliran kode membantu pengembang untuk mengikuti logika yang diterapkan dalam aplikasi dan memastikan bahwa semua jalur eksekusi yang relevan telah dipertimbangkan dan diuji.

9. Error Codes

- **Definisi:** Elemen ini mendeskripsikan kode kesalahan yang digunakan dalam penanganan kesalahan di dalam solusi. Kode ini biasanya terdiri dari angka atau string yang mewakili jenis kesalahan tertentu.
- **Peran:** Dengan menggunakan kode kesalahan yang jelas, tim dapat lebih mudah mengidentifikasi masalah ketika kesalahan terjadi. Ini juga memudahkan dokumentasi dan pelaporan kesalahan kepada pengguna atau pengembang lain.

BAB 11
DESAIN LAPISAN DATA

11.1 MENDESAIN PENYIMPANAN DATA

Dalam *Microsoft Solution Framework* (MSF), mendesain penyimpanan data merupakan elemen penting dari desain lapisan data dalam rekayasa perangkat lunak. Lapisan data adalah bagian aplikasi yang bertanggung jawab untuk mengelola, mengakses, dan menyimpan data aplikasi ke dalam database atau penyimpanan lain yang relevan. Mendesain lapisan ini melibatkan beberapa langkah dan konsep kunci yang bertujuan memastikan efisiensi, keamanan, dan kemudahan akses data dalam aplikasi.

Fase planning

Dalam fase planning (perencanaan) dari *Microsoft Solution Framework* (MSF), tim proyek bertugas mendesain lapisan data yang menjadi fondasi penyimpanan dan pengelolaan data pada aplikasi yang akan dikembangkan. Lapisan data dalam solusi ini terbagi menjadi dua komponen utama, yaitu penyimpanan data dan layanan data. Kedua komponen ini berperan penting dalam memastikan data aplikasi dikelola dengan aman, terstruktur, dan mudah diakses saat dibutuhkan oleh komponen lainnya.

1. Penyimpanan Data

Penyimpanan data bertanggung jawab atas pengorganisasian dan penyimpanan informasi secara efisien. Penyimpanan data umumnya bergantung pada sistem basis data (database) yang digunakan untuk menyimpan, mengelola, dan mengakses informasi. Berikut adalah beberapa elemen kunci dalam desain penyimpanan data:

- **Basis Data**: Basis data merupakan sistem yang dirancang untuk mengelola dan menyimpan data aplikasi. Data yang disimpan di sini dapat diakses dan dikelola melalui query (perintah) yang diatur dengan bahasa tertentu, seperti SQL (Structured Query Language) untuk database relasional.
- **Entitas Data**: Setiap data dalam basis data memiliki entitas, yaitu objek yang menyimpan informasi khusus dan relevan untuk aplikasi. Contoh entitas bisa berupa "Pelanggan" dalam sistem manajemen pelanggan, yang memiliki atribut seperti nama, alamat, dan nomor kontak.
- **Skema Data**: Merupakan desain struktur dari basis data, di mana tabel, kolom, relasi, serta indeks dirancang untuk memastikan data tersimpan secara efisien dan mendukung kebutuhan aplikasi.

2. Layanan Data

Layanan data adalah komponen yang menyediakan akses data kepada lapisan-lapisan aplikasi lainnya. Layanan ini berfungsi sebagai perantara antara aplikasi dan basis data, serta bertanggung jawab untuk menyediakan fungsi manajemen data, seperti *Create*, *Read*, *Update*, dan *Delete* (CRUD). Layanan data juga mengimplementasikan keamanan, integritas, dan optimasi data. Berikut adalah aspek penting dalam desain layanan data:

- ✓ **Data Access Layer (DAL)**: DAL adalah bagian yang mengelola interaksi langsung dengan basis data, memungkinkan aplikasi untuk mengambil atau memperbarui data secara aman dan efisien. DAL dapat menggunakan objek data yang mapan (ORM) untuk memperlancar akses ke database.
- ✓ **Transaksi Data**: Layanan data memastikan bahwa semua transaksi dilakukan secara konsisten, sehingga operasi yang melibatkan beberapa tahap (misalnya, pemindahan uang antar akun) dapat dieksekusi dengan benar tanpa gangguan.
- ✓ **Keamanan Akses Data**: Mengatur dan mengendalikan siapa saja yang memiliki izin untuk mengakses atau mengubah data. Ini dapat mencakup enkripsi data, otorisasi pengguna, dan pengaturan peran pengguna

11.2 KOMPONEN-KUNCI DALAM DESAIN PENYIMPANAN DAN LAYANAN DATA

Untuk mendukung aplikasi yang aman dan efisien, beberapa komponen penting perlu diterapkan dalam desain lapisan data:

- ❖ **Normalisasi Data**: Penyimpanan data sering kali memerlukan proses normalisasi, yaitu teknik yang digunakan untuk mengorganisir tabel dan kolom dalam basis data agar menghindari redundansi dan meningkatkan efisiensi.
- ❖ **Indeks dan Optimasi Query**: Indeks digunakan untuk mempercepat pencarian data. Namun, jumlah dan tipe indeks perlu diatur dengan baik untuk menjaga kinerja sistem.
- ❖ **Cache Data**: Layanan data dapat menggunakan caching untuk menyimpan data yang sering diakses, mengurangi beban pada basis data utama dan mempercepat waktu respons.
- ❖ **Backup dan Recovery**: Penyimpanan data perlu diikuti dengan mekanisme pencadangan dan pemulihan (backup and recovery) untuk mengantisipasi risiko kehilangan data akibat kesalahan sistem atau bencana.
- ❖ **Logging dan Monitoring**: Melacak operasi data dan memonitor akses membantu mengidentifikasi masalah serta memastikan keamanan dan kinerja data.

Basis data

Adalah sistem yang menyimpan kumpulan nilai data yang diatur secara terstruktur. Data ini disusun menggunakan skema yang mendefinisikan bagaimana data diatur, diakses, dan dikelola dalam basis data tersebut. Struktur yang baik memungkinkan basis data untuk mendukung aplikasi dalam mengelola informasi secara efisien, sehingga mempermudah proses bisnis atau penggunaan lain dalam suatu sistem.

Struktur Skema Basis Data

Skema basis data adalah inti dari perencanaan basis data, yang menentukan cara data diatur secara internal. Dalam skema, terdapat elemen-elemen seperti entitas, atribut, dan relasi antar entitas yang menciptakan struktur data yang jelas.

1. **Entitas**: Objek utama dalam basis data yang merepresentasikan objek nyata atau konsep bisnis. Misalnya, dalam sistem manajemen perpustakaan, entitas bisa berupa "Buku," "Anggota," atau "Transaksi."

2. **Atribut**: Properti atau karakteristik dari entitas yang menyimpan informasi spesifik. Misalnya, entitas "Buku" mungkin memiliki atribut seperti "Judul," "Pengarang," "ISBN," dan "Tanggal Terbit."
3. **Relasi Antar Entitas**: Hubungan antara entitas yang berbeda, yang membantu menghubungkan data dalam konteks bisnis. Misalnya, relasi antara "Anggota" dan "Transaksi" menunjukkan hubungan anggota yang meminjam buku.

Desain Logikal dan Fisikal Basis Data

Proses desain basis data melibatkan dua tahap utama: desain logikal dan desain fisikal.

1. **Desain Logikal**: Dalam tahap ini, tim proyek membuat gambaran konseptual dari basis data tanpa memperhatikan aspek teknis penyimpanannya. Desain logikal menggambarkan entitas, atribut, dan hubungan antar entitas melalui diagram, seperti *Entity-Relationship Diagram* (ERD). Di sini, fokusnya adalah memastikan semua elemen bisnis direpresentasikan dan hubungan antar elemen bisnis dijelaskan dengan baik.
2. **Desain Fisikal**: Pada tahap ini, tim proyek mengimplementasikan skema ke dalam struktur fisik yang sesuai dengan perangkat lunak basis data yang akan digunakan, seperti MySQL, PostgreSQL, atau Oracle. Desain fisikal mencakup pemilihan tipe data untuk setiap atribut, pengaturan indeks untuk mempercepat akses data, dan skema penyimpanan untuk operasi CRUD (*Create, Read, Update, Delete*). Tujuan dari desain fisikal adalah menciptakan basis data yang efisien dan optimal untuk mendukung performa aplikasi.

Implementasi dan Manfaat Basis Data

Implementasi basis data yang dirancang dengan baik memungkinkan aplikasi untuk menyimpan dan mengelola data dengan lancar dan terstruktur, mendukung analisis, pelacakan informasi, dan manajemen data yang akurat. Basis data yang baik mempercepat pengambilan data dan memudahkan pengguna dalam mengekstraksi informasi yang dibutuhkan. Selain itu, pemeliharaan basis data yang baik membantu mengurangi kesalahan dan memastikan integritas data tetap terjaga.

Entitas dan Atribut dalam Basis Data

Dalam basis data, *entitas* merupakan objek yang memiliki keberadaan nyata atau konsep yang perlu didefinisikan dan diwakili dalam sistem informasi. Entitas sering kali mencerminkan elemen penting dalam proses bisnis, seperti "Produk," "Order," atau "Pelanggan." Entitas berperan sebagai wadah penyimpanan informasi, yang memungkinkan sistem untuk mengelola dan menghubungkan berbagai data secara efisien.

Entitas dan Atribut

Entitas adalah komponen kunci dalam desain basis data karena merepresentasikan objek nyata atau abstrak yang akan disimpan dalam sistem. Identifikasi entitas menjadi langkah awal dalam desain logikal basis data, yang bertujuan untuk menangkap elemen-elemen yang relevan dengan kebutuhan bisnis atau aplikasi. Misalnya, dalam sistem penjualan, entitas yang mungkin diidentifikasi meliputi:

- **Produk**: Menyimpan informasi tentang barang yang dijual.
- **Order**: Mencatat pesanan yang dibuat oleh pelanggan.

- **Pelanggan**: Menyimpan data pelanggan yang melakukan transaksi.

Pada tahap desain, pengembang akan memetakan entitas-entitas ini untuk memastikan semua aspek yang diperlukan dalam proses bisnis tercakup.

Konsep Atribut

- *Atribut* adalah karakteristik atau properti yang memberikan rincian lebih lanjut mengenai suatu entitas. Atribut dapat dianggap sebagai kolom atau field dalam tabel basis data, yang menjelaskan entitas secara lebih spesifik. Setiap entitas memiliki beberapa atribut yang menyimpan informasi terkait, misalnya:

  - ✓ Untuk entitas **Produk**, atributnya bisa meliputi "Nama Produk," "Harga," dan "Stok."
  - ✓ Untuk entitas **Order**, atributnya bisa berupa "Tanggal Order," "Jumlah Barang," dan "Total Harga."
  - ✓ Untuk entitas **Pelanggan**, atributnya dapat berupa "Nama Pelanggan," "Alamat," dan "Nomor Telepon."

Identifikasi Entitas dan Atribut

Proses identifikasi entitas dan atribut adalah langkah kritis dalam desain logikal basis data. Ini melibatkan:

- **Analisis Kebutuhan Bisnis**: Menentukan objek atau konsep mana yang akan menjadi bagian dari sistem.
- **Pemetaan Entitas**: Membuat daftar entitas yang relevan dengan proses bisnis.
- **Penentuan Atribut untuk Setiap Entitas**: Mengidentifikasi rincian penting dari setiap entitas, seperti harga produk atau tanggal pesanan.

Proses ini membantu memastikan bahwa sistem memiliki data yang akurat dan lengkap untuk mendukung fungsionalitas yang dibutuhkan.

Pentingnya Hubungan antara Entitas

Setelah entitas dan atribut diidentifikasi, langkah selanjutnya adalah mendefinisikan hubungan antara entitas. Hubungan ini penting untuk menghubungkan entitas yang memiliki kaitan dalam konteks bisnis. Misalnya, entitas *Order* dapat dihubungkan dengan entitas *Pelanggan* untuk melacak pelanggan yang melakukan transaksi tertentu. Hubungan ini membuat basis data lebih dinamis dan memungkinkan pengguna atau aplikasi untuk memperoleh informasi yang lebih komprehensif

Pada sebuah *use case*

kita dapat mengidentifikasi objek-objek utama yang berperan dalam skenario proses bisnis. Objek-objek ini akan menjadi dasar untuk mengembangkan entitas dalam basis data, karena mereka mewakili elemen kunci yang harus dikelola dan dihubungkan dalam sistem. Misalnya, dalam *use case* “karyawan membuat sebuah kontrak dengan pelanggan,” kita mengenali tiga objek penting: Karyawan, Kontrak, dan Pelanggan. Masing-masing objek ini memiliki peran spesifik:

1. **Karyawan**: Bertindak sebagai pelaku utama yang melakukan aksi di dalam sistem. Dalam konteks ini, karyawan yang berinteraksi dengan pelanggan mencerminkan tanggung jawab perusahaan untuk mengelola kesepakatan bisnis, memberikan

pelayanan, atau menjual produk. Dalam basis data, entitas *Karyawan* akan menyimpan informasi seperti nama karyawan, jabatan, nomor identifikasi, dan lainnya yang relevan untuk mengidentifikasi serta melacak aktivitas setiap karyawan dalam sistem.

2. **Pelanggan**: Pelanggan berperan sebagai penerima aksi, yang berhubungan langsung dengan karyawan untuk membuat kesepakatan atau kontrak. Entitas *Pelanggan* penting untuk menyimpan informasi yang memungkinkan perusahaan memahami kebutuhan pelanggan, riwayat transaksi, atau interaksi yang telah dilakukan. Data pelanggan, seperti nama, kontak, alamat, dan preferensi bisnis, sering kali disimpan sebagai atribut di dalam basis data untuk memudahkan manajemen relasi dengan pelanggan.
3. **Kontrak**: Objek *Kontrak* menggambarkan kesepakatan atau perjanjian formal antara karyawan dan pelanggan. Dalam basis data, *Kontrak* berfungsi sebagai entitas yang menyimpan rincian terkait dengan persetujuan, seperti tanggal mulai, ketentuan kontrak, produk atau layanan yang disetujui, serta status kontrak (aktif, selesai, atau dibatalkan). Kontrak menjadi penghubung antara karyawan dan pelanggan, memungkinkan kedua pihak untuk memiliki panduan yang jelas dalam menjalankan bisnis.

Implementasi dalam Basis Data

Di dalam basis data, setiap objek dari *use case* ini diimplementasikan sebagai entitas dengan atribut dan relasi yang menggambarkan keterkaitan mereka. Dalam skema basis data, relasi antar entitas mungkin seperti berikut:

- *Karyawan* berelasi dengan *Kontrak* melalui atribut seperti *ID Karyawan* pada entitas *Kontrak*, yang mengidentifikasi siapa yang membuat kontrak.
- *Pelanggan* berelasi dengan *Kontrak* melalui atribut *ID Pelanggan*, yang mengidentifikasi dengan siapa kontrak dilakukan.

Manfaat Identifikasi Objek dalam Use Case

Dengan mengenali objek-objek ini, pengembang dan perancang basis data dapat membangun struktur data yang lebih terfokus pada kebutuhan bisnis. Selain itu, identifikasi objek pada tahap awal memungkinkan tim untuk:

- Menyusun skema basis data yang jelas dan memenuhi kebutuhan operasi bisnis,
- Mempersiapkan data yang lengkap untuk mendukung aplikasi dan laporan,
- Menyediakan akses data yang cepat dan relevan bagi karyawan, pelanggan, dan pengambil keputusan di dalam organisasi.

Definisi Atribut

Setelah mengidentifikasi entitas dalam suatu sistem, langkah berikutnya adalah mendefinisikan atribut. Atribut merupakan karakteristik atau ciri yang menjelaskan entitas secara lebih mendetail. Berikut adalah penjelasan yang lebih mendalam mengenai atribut dan karakteristiknya:

1. Deskripsi Entitas

Atribut memberikan informasi spesifik mengenai entitas tersebut. Sebagai contoh, dalam entitas "karyawan," atribut yang dapat didefinisikan antara lain:

- **no_induk**: Nomor identifikasi unik yang diberikan kepada setiap karyawan.
- **nama**: Nama lengkap karyawan.
- **alamat**: Lokasi tempat tinggal karyawan.
- **departemen**: Bagian atau unit tempat karyawan bekerja.

Setiap atribut ini membantu untuk mendeskripsikan dan mengidentifikasi karyawan dalam konteks yang lebih luas.

2. Kehadiran Atribut

Atribut hanya dianggap "ada" jika ia dicantumkan dan diterapkan dalam konteks entitas tertentu. Contohnya, atribut seperti "warna" tidak akan memiliki makna atau relevansi jika tidak diterapkan pada objek yang sesuai. Misalnya, jika kita berbicara tentang entitas "mobil," atribut "warna" baru akan memiliki arti ketika kita mendefinisikan jenis warna yang diaplikasikan pada mobil tertentu.

3. Definisi Kolom Tabel dalam Basis Data

Atribut dalam basis data mendefinisikan struktur tabel dan menentukan karakteristik dari setiap kolom dalam tabel tersebut. Setiap atribut memiliki tipe data yang jelas, seperti integer, string, atau tanggal, yang menentukan jenis nilai yang dapat disimpan dalam kolom tersebut. Selain itu, atribut juga memiliki ukuran data yang mengatur seberapa banyak memori yang diperlukan untuk menyimpan nilai, seperti menentukan jumlah maksimum karakter untuk atribut "nama." Tidak hanya itu, atribut juga dapat memiliki hubungan dengan atribut lain, baik dalam tabel yang sama maupun tabel lain. Sebagai contoh, atribut "departemen" pada entitas "karyawan" dapat merujuk ke entitas lain seperti "departemen," yang kemudian menjelaskan lebih lanjut mengenai informasi seperti "nama departemen" dan "lokasi."

4. Contoh dalam Konteks Basis Data

Dalam suatu basis data relasional, misalkan kita memiliki dua entitas: Karyawan dan Departemen. Tabel "Karyawan" dapat memiliki atribut sebagai berikut:

- ✓ no_induk (integer, 10)
- ✓ nama (varchar, 100)
- ✓ alamat (varchar, 255)
- ✓ departemen_id (integer, merujuk ke tabel Departemen)

Tabel "Departemen" dapat memiliki atribut:

- ✓ id (integer, 10)
- ✓ nama_departemen (varchar, 100)
- ✓ lokasi (varchar, 100)

Dalam hal ini, atribut "departemen_id" di tabel "Karyawan" menghubungkan karyawan dengan departemen mereka, menciptakan relasi antar tabel.

Tabel dan Kolom dalam Basis Data Relasional

Tabel merupakan representasi dari sebuah entitas dalam database relasional. Setiap tabel dirancang untuk menyimpan data yang beragam sesuai dengan kebutuhan bisnis, seperti informasi pelanggan, produk, atau transaksi. Tabel memiliki struktur yang terorganisir dan memudahkan dalam pengelolaan dan pencarian data.

Struktur Tabel:

Tabel terdiri dari beberapa elemen, yaitu:

- **Baris (Row):** Mewakili satu record atau item data. Setiap baris berisi informasi lengkap tentang entitas tertentu.
- **Kolom (Column):** Mewakili atribut dari entitas. Setiap kolom menyimpan jenis data yang sama untuk setiap record dalam tabel.

3. Penyimpanan Data:

Data dalam tabel disimpan dalam bentuk baris. Setiap baris menyimpan informasi yang berbeda-beda, tetapi mengikuti format yang telah ditentukan oleh kolom. Ini memudahkan pemrosesan dan pengambilan data.

4. Keunikan Record:

Setiap record dalam tabel haruslah unik. Keunikan ini biasanya dijamin dengan menggunakan kunci primer (primary key). Kunci primer adalah kolom atau kombinasi kolom yang secara unik mengidentifikasi setiap record dalam tabel. Misalnya, kolom "no induk" dalam tabel "Karyawan" dapat berfungsi sebagai kunci primer.

5. Manfaat Tabel:

Tabel memungkinkan organisasi untuk mengelola data dengan cara yang terstruktur. Data dapat diakses, diperbarui, dan dihapus dengan lebih mudah. Selain itu, tabel juga memfasilitasi pembuatan relasi antar entitas, yang penting dalam basis data yang kompleks.

Kolom (Field) dalam Tabel

1. Definisi Kolom:

Kolom, atau field, adalah bagian dari tabel yang menyimpan nilai data untuk setiap record. Setiap kolom memiliki nama yang mendeskripsikan data yang disimpan dan memiliki tipe data tertentu.

2. Tipe Data Kolom:

Setiap kolom dalam tabel memiliki tipe data yang menentukan jenis nilai yang dapat disimpan. Tipe data umum meliputi:

- **Integer:** Untuk menyimpan angka bulat.
- **Varchar:** Untuk menyimpan string atau teks dengan panjang variabel.
- **Date:** Untuk menyimpan nilai tanggal.
- **Boolean:** Untuk menyimpan nilai benar/salah (true/false).

3. Ukuran dan Batasan Kolom:

Kolom juga dapat memiliki ukuran tertentu yang membatasi seberapa banyak data yang dapat disimpan. Misalnya, kolom "nama" mungkin dibatasi hingga 100 karakter. Selain itu, batasan lain seperti NOT NULL dapat diterapkan untuk memastikan bahwa kolom tertentu tidak boleh kosong.

4. Nilai yang Berbeda dalam Kolom:

Setiap kolom menyimpan nilai yang berbeda untuk setiap record dalam tabel. Sebagai contoh, dalam tabel "Karyawan," kolom "nama" akan berisi nama yang berbeda untuk setiap karyawan. Ini memungkinkan variasi dan detail informasi yang dapat dikelola dengan baik.

5. Relasi Antar Kolom:

Kolom dapat memiliki hubungan dengan kolom dalam tabel lain. Ini sering digunakan untuk membangun relasi antar tabel dalam database. Misalnya, kolom "departemen_id" dalam tabel "Karyawan" dapat merujuk ke kolom "id" dalam tabel "Departemen," membentuk hubungan antara karyawan dan departemen mereka.

Kunci (Key)

Kunci dalam model data berfungsi untuk mengidentifikasi secara unik setiap contoh (instance) dari entitas, memastikan bahwa setiap catatan atau data dalam sistem memiliki identitas yang jelas dan dapat ditelusuri. Kunci ini sangat penting untuk mengelola relasi antar entitas dalam sebuah basis data, khususnya dalam skema relasional. Secara umum, kunci terdiri dari dua jenis utama: Kunci Primer (Primary Key) dan Kunci Tamu (Foreign Key).

Jenis Kunci dalam Basis Data

1. **Kunci Primer (Primary Key)**
   - **Definisi**: Kunci primer adalah atribut atau kumpulan atribut yang secara unik mengidentifikasi setiap baris dalam sebuah tabel. Artinya, tidak ada dua baris yang dapat memiliki nilai yang sama dalam kolom yang ditentukan sebagai kunci primer. Kunci primer adalah fondasi utama dalam menentukan identitas data pada tingkat entitas.
   - **Contoh**: Misalnya, dalam tabel *SalesOrder*, atribut *SalesOrderId* dijadikan kunci primer karena nilainya unik untuk setiap transaksi order yang dilakukan. Dengan demikian, setiap order dapat diakses dan diidentifikasi dengan jelas menggunakan *SalesOrderId*, yang mencegah duplikasi dan memastikan integritas data.
   - **Karakteristik**: Kunci primer harus selalu memiliki nilai (tidak boleh *null*), dan nilainya harus unik. Kunci primer biasanya merupakan data numerik sederhana atau kode alfanumerik yang memungkinkan data diakses dengan lebih cepat.
2. **Kunci Tamu (Foreign Key)**
   - **Definisi**: Kunci tamu adalah kunci yang menghubungkan dua tabel dalam basis data, menciptakan relasi antar entitas dengan mengacu pada kunci primer tabel lain. Kunci tamu biasanya merupakan atribut dalam satu tabel yang mengacu pada kunci primer di tabel lain, membangun koneksi antara data yang saling terkait.
   - **Contoh**: Dalam tabel *SalesOrderDetail*, *ProductID* berfungsi sebagai kunci tamu yang menghubungkan data order dengan informasi produk yang dijual. Di tabel *MasterProduct*, *ProductID* adalah kunci primer yang menyimpan informasi produk utama seperti nama produk, harga, dan stok. Ketika *ProductID* muncul di *SalesOrderDetail*, ia berfungsi sebagai kunci tamu yang menghubungkan detail penjualan dengan produk yang dipesan.
   - **Karakteristik**: Kunci tamu tidak harus unik karena dapat muncul lebih dari satu kali dalam tabel, terutama saat ada banyak referensi ke satu data di tabel yang terkait. Namun, kunci tamu tidak boleh bernilai *null* jika berfungsi sebagai bagian dari relasi yang wajib.

Fungsi dan Manfaat Penggunaan Kunci

Penggunaan kunci primer dan kunci tamu dalam desain basis data memberikan manfaat yang signifikan, termasuk:

- **Integritas Data**: Dengan menggunakan kunci primer dan kunci tamu, basis data dapat mencegah terjadinya data ganda dan menjaga konsistensi antar tabel. Misalnya, jika ada pesanan di *SalesOrderDetail*, harus ada produk terkait yang sesuai di *MasterProduct*.
- **Hubungan Antar Tabel**: Kunci tamu memungkinkan relasi antara tabel, membuat basis data lebih terstruktur dan memudahkan pencarian informasi antar entitas. Relasi ini memungkinkan pelacakan data secara komprehensif dan efisien.
- **Kecepatan Akses Data**: Kunci primer mempercepat proses pengambilan data karena setiap data diidentifikasi dengan atribut yang unik, mempermudah pencarian dan pengelolaan data dalam skala besar.

Implementasi dalam Desain Basis Data

Implementasi kunci primer dan kunci tamu biasanya dilakukan dalam tahap desain logikal dan fisik basis data:

- **Desain Logikal**: Pada tahap ini, tim proyek mendefinisikan entitas, atribut, dan relasi dasar dalam model data. Kunci primer dipilih untuk entitas, dan relasi antar entitas dirancang menggunakan kunci tamu.
- **Desain Fisik**: Pada tahap ini, tim menentukan tipe data dan sifat atribut, mengimplementasikan kunci primer dan tamu pada tabel basis data. Desain fisik termasuk memastikan kunci primer bersifat unik dan kunci tamu mengacu pada kunci primer yang benar.

Poin Utama tentang Kunci dalam Basis Data

- **Kunci Primer**: Identifikasi unik di setiap tabel, tidak boleh *null*, dan harus unik.
- **Kunci Tamu**: Menghubungkan dua tabel, mengacu pada kunci primer, dan mendukung integrasi data.

Dengan mendesain kunci secara efektif, basis data tidak hanya menjadi lebih terstruktur tetapi juga lebih efisien, memfasilitasi pengelolaan data yang lebih baik dan memungkinkan aplikasi untuk mengakses data dengan akurat dan cepat.

Relasi

Relasi antar entitas dalam basis data memungkinkan tabel-tabel saling terhubung, memfasilitasi penyimpanan dan akses data yang lebih terstruktur. Relasi ini diwujudkan dengan menambahkan kunci dari satu entitas (tabel) sebagai atribut di entitas lain yang berhubungan. Kunci yang ditambahkan ini disebut **kunci tamu (foreign key),** dan relasi yang tercipta antara dua tabel memungkinkan basis data menghubungkan dan mengelola data yang terpisah secara efisien.

Jenis-Jenis Relasi Antar Entitas

1. **Relasi Satu-ke-Satu (One-to-One)**
   - **Definisi**: Dalam relasi satu-ke-satu, setiap instance dari entitas pertama memiliki hubungan dengan tepat satu instance dari entitas kedua, dan sebaliknya.

- **Contoh**: Misalnya, dalam sistem informasi HRD, tabel *Employee* dan *EmployeeDetails* dapat memiliki relasi satu-ke-satu jika setiap pegawai hanya memiliki satu catatan detail pribadi, seperti alamat rumah atau data keluarga.
- **Implementasi**: Relasi ini sering kali digunakan untuk memisahkan data sensitif atau data yang jarang diakses ke dalam tabel terpisah. Implementasinya dilakukan dengan menambahkan kunci primer dari satu tabel sebagai kunci tamu yang unik di tabel lain.

2. **Relasi Satu-ke-Banyak (One-to-Many)**
   - **Definisi**: Dalam relasi satu-ke-banyak, setiap instance dari entitas pertama dapat memiliki hubungan dengan banyak instance dari entitas kedua, tetapi setiap instance dari entitas kedua hanya terkait dengan satu instance dari entitas pertama.
   - **Contoh**: Sebagai contoh, dalam tabel *Customer* dan *Order*, satu pelanggan dapat memiliki banyak order, tetapi setiap order hanya terhubung dengan satu pelanggan.
   - **Implementasi**: Relasi ini dibangun dengan menambahkan kunci primer dari tabel *Customer* sebagai kunci tamu di tabel *Order*. Ini berarti bahwa setiap order akan menyimpan ID pelanggan, memungkinkan identifikasi pelanggan yang terkait dengan setiap order.
3. **Relasi Banyak-ke-Banyak (Many-to-Many)**
   - **Definisi**: Dalam relasi banyak-ke-banyak, setiap instance dari entitas pertama dapat memiliki hubungan dengan banyak instance dari entitas kedua, dan sebaliknya.
   - **Contoh**: Sebagai contoh, dalam sistem manajemen kursus, tabel *Student* dan *Course* dapat memiliki relasi banyak-ke-banyak, karena setiap siswa bisa terdaftar di beberapa kursus, dan setiap kursus bisa memiliki beberapa siswa.
   - **Implementasi**: Relasi ini diwujudkan dengan membuat tabel antara, yang sering disebut *junction table* atau tabel perantara. Misalnya, tabel *StudentCourse* dapat dibuat, berisi kunci primer dari tabel *Student* dan *Course* sebagai kunci tamu. Dengan demikian, setiap record di tabel perantara akan menghubungkan satu siswa dengan satu kursus.

Manfaat dan Pentingnya Relasi Antar Entitas

1. **Integrasi dan Konsistensi Data**: Relasi antar entitas memungkinkan integrasi yang lebih baik antara tabel yang terpisah. Dengan menghubungkan data melalui kunci tamu, basis data dapat mempertahankan konsistensi antar entitas, sehingga meminimalisir data ganda dan ketidakcocokan.
2. **Kemudahan Pengelolaan Data**: Relasi membantu mengelompokkan data yang relevan dalam tabel terpisah. Misalnya, dengan memisahkan tabel *Order* dan *Customer*, basis data dapat menangani data order dan pelanggan secara efisien. Relasi yang terbentuk memungkinkan pengguna untuk menampilkan semua order untuk pelanggan tertentu atau sebaliknya.

3. **Optimasi Query dan Kecepatan Akses**: Dengan membangun relasi yang efisien, basis data dapat mempercepat waktu akses karena pengambilan data spesifik yang terkait. Misalnya, query untuk mengambil semua produk dari order tertentu bisa dijalankan dengan cepat melalui relasi antara tabel *Order* dan *Product*.
4. **Mempermudah Pemeliharaan Basis Data**: Pemeliharaan data juga lebih mudah dengan relasi yang terstruktur. Misalnya, ketika seorang pelanggan dihapus dari tabel *Customer*, semua data terkait (seperti order dalam tabel *Order*) bisa dihapus otomatis atau diperbarui menggunakan konsep *referential integrity*, jika fitur tersebut diaktifkan.

Implementasi Relasi dalam Proses Desain Basis Data

Proses desain basis data melibatkan beberapa langkah dalam merancang relasi antar entitas, yaitu:

- **Identifikasi Entitas dan Relasi**: Tim proyek pertama-tama mengidentifikasi entitas utama dalam sistem (misalnya, *Customer*, *Product*, dan *Order*), lalu menentukan relasi antar entitas (misalnya, *Customer* memiliki banyak *Order*).
- **Penambahan Kunci Tamu**: Untuk mengimplementasikan relasi, kunci tamu ditambahkan pada tabel yang berhubungan. Misalnya, *CustomerID* dari tabel *Customer* ditambahkan sebagai kunci tamu di tabel *Order* untuk menunjukkan bahwa setiap order terkait dengan pelanggan tertentu.
- **Membangun *Junction Table* untuk Relasi Banyak-ke-Banyak**: Jika ada relasi banyak-ke-banyak, tabel perantara dibuat untuk mengelola relasi tersebut. Misalnya, tabel *Enrollment* bisa dibuat untuk menghubungkan tabel *Student* dan *Course*.
- **Uji dan Validasi**: Tim melakukan uji coba untuk memastikan relasi telah diterapkan dengan benar. Tes ini melibatkan query antar tabel untuk memastikan bahwa data terhubung secara akurat dan konsisten.

Poin-Poin Utama Relasi Antar Entitas

- **Relasi Satu-ke-Satu**: Hubungan unik antar dua tabel, digunakan untuk data yang jarang digunakan atau data sensitif.
- **Relasi Satu-ke-Banyak**: Hubungan satu entitas dengan beberapa entitas lain, misalnya pelanggan dan order.
- **Relasi Banyak-ke-Banyak**: Hubungan yang membutuhkan tabel perantara untuk menghubungkan banyak instance antar entitas, seperti siswa dan kursus.
- **Penggunaan Kunci Tamu**: Untuk menciptakan hubungan antar entitas yang saling terkait.

Dengan menerapkan relasi antar entitas, basis data menjadi lebih terstruktur dan mudah dipelihara, memungkinkan pengelolaan data dalam jumlah besar secara efisien serta memastikan bahwa sistem dapat memberikan informasi secara akurat dan responsif kepada pengguna.

Mengoptimalkan akses data

Mengoptimalkan akses data adalah proses mempercepat dan meningkatkan efisiensi pengambilan, penyimpanan, dan pembaruan data dalam sistem basis data. Tujuan utamanya

adalah memastikan aplikasi dapat memberikan respons yang cepat terhadap permintaan data, terutama dalam lingkungan dengan volume data besar atau pengguna yang banyak.Optimasi akses data mencakup berbagai teknik, seperti pengindeksan untuk mempercepat pencarian data spesifik, perancangan query yang efisien untuk meminimalkan beban pemrosesan, dan pembagian tabel besar menjadi bagian-bagian yang lebih kecil (partisi) agar proses pencarian lebih fokus dan cepat. Selain itu, mengatur struktur basis data dengan teknik normalisasi dan denormalisasi dapat mengurangi redundansi dan mempercepat akses data, sedangkan caching menyimpan data sementara di memori untuk diambil lebih cepat tanpa mengakses basis data utama. Intinya, optimasi akses data memungkinkan sistem mengelola data dengan efisien sehingga aplikasi lebih cepat dalam memproses dan menampilkan data bagi pengguna.

Untuk mengoptimalkan basis data, berbagai teknik dapat diterapkan guna meningkatkan efisiensi dan performa dalam pengaksesan dan pembaruan data. Teknik optimasi ini sangat penting, terutama bagi aplikasi dengan volume data besar atau yang membutuhkan waktu akses cepat. Salah satu teknik dasar dalam optimasi basis data adalah pengindeksan, yang dapat mempercepat pencarian dan pengurutan data dalam tabel.

Teknik Pengindeksan dalam Basis Data

Pengindeksan adalah proses membuat struktur tambahan pada basis data yang memungkinkan sistem mengakses data lebih cepat. Tanpa indeks, setiap kali data dicari atau diurutkan, sistem harus memeriksa setiap baris satu per satu, yang tentunya tidak efisien. Indeks memungkinkan DBMS (Database Management System) untuk menemukan data secara cepat tanpa harus menelusuri seluruh tabel.

Jenis-jenis Indeks dalam Basis Data:

1. **Clustered Indeks**
    a. **Pengurutan Fisik Data**: Clustered indeks melakukan pengurutan ulang data secara fisik dalam tabel agar sesuai dengan urutan indeks. Ini berarti bahwa data disusun langsung pada penyimpanan dalam urutan tertentu.
    b. **Kinerja yang Tinggi untuk Pembacaan**: Clustered indeks memberikan kinerja yang sangat baik dalam operasi pembacaan data, terutama dalam situasi di mana data dibaca secara berurutan atau dalam volume besar. Karena data sudah diurutkan, operasi pencarian, pembaruan, dan penghapusan dapat dilakukan lebih efisien.
    c. **Pembatasan Satu Indeks**: Setiap tabel hanya boleh memiliki satu clustered indeks karena fisik datanya sudah diatur ulang sesuai dengan urutan yang diinginkan oleh indeks.
2. **Nonclustered Indeks**
    a. **Penyimpanan Indeks Terpisah**: Nonclustered indeks tidak mengubah urutan fisik data dalam tabel. Sebaliknya, ia menyimpan indeks dalam struktur terpisah, yang menunjuk ke lokasi data yang relevan dalam tabel.
    b. **Dapat Memiliki Lebih dari Satu Indeks**: Tidak seperti clustered indeks, sebuah tabel dapat memiliki beberapa nonclustered indeks. Hal ini memungkinkan optimasi pencarian yang lebih fleksibel untuk berbagai kolom dalam tabel.

c. **Fokus pada Kolom Tertentu**: Nonclustered indeks mengelola informasi indeks dari kolom atau kelompok kolom yang tidak harus berhubungan langsung dengan urutan data secara keseluruhan. Ini berguna untuk operasi pencarian dan filter data pada kolom yang sering digunakan dalam query.

Keuntungan dan Tantangan dalam Penggunaan Indeks

- **Keuntungan**: Indeks mempercepat pencarian dan pengurutan data sehingga meningkatkan performa secara signifikan dalam operasi pembacaan dan pengambilan data.
- **Tantangan**: Indeks membutuhkan penyimpanan tambahan dalam basis data. Jika ada terlalu banyak indeks, operasi pembaruan dan penyisipan data mungkin melambat karena setiap perubahan data memerlukan pemutakhiran indeks.

Pemartisan data

Pemartisian data (data partitioning) adalah teknik yang dapat membantu meningkatkan kecepatan akses data dalam basis data dengan cara membagi data menjadi segmen-segmen yang lebih kecil dan mudah dikelola. Dengan pemartisian, beban kerja pada basis data dapat tersebar lebih merata, sehingga operasi pengaksesan dan pemrosesan data dapat dilakukan lebih efisien. Teknik ini sangat berguna dalam skenario dengan jumlah data besar atau basis data yang sering diakses oleh banyak pengguna secara bersamaan.

Jenis-jenis Pemartisian Data

Pemartisian data pada umumnya terbagi menjadi dua jenis utama, yaitu:

1. **Partisi Horizontal**
   - **Konsep Partisi Horizontal**: Dalam partisi horizontal, data dalam tabel dibagi berdasarkan baris (record). Tabel utama dipecah menjadi beberapa tabel yang masing-masing menyimpan sebagian dari record yang sama, namun tetap memiliki kolom-kolom yang identik dengan tabel aslinya. Hal ini memungkinkan tabel yang dihasilkan menyimpan sejumlah record yang berbeda tetapi tetap memiliki struktur yang sama.
   - **Contoh Partisi Horizontal**: Misalnya, tabel transaksi pada basis data penjualan dapat dipartisi menjadi dua tabel berdasarkan periode waktu, seperti Transaksi_Jan-Jun dan Transaksi_Jul-Des. Kedua tabel ini memiliki struktur kolom yang sama, tetapi menyimpan data transaksi pada periode yang berbeda. Dengan cara ini, aplikasi dapat mengakses data transaksi lebih cepat, karena hanya perlu membaca sebagian dari keseluruhan data transaksi.
2. **Partisi Vertikal**
   - **Konsep Partisi Vertikal**: Dalam partisi vertikal, data dibagi berdasarkan kolom. Tabel utama dipecah menjadi beberapa tabel yang menyimpan sebagian kolom, tetapi masing-masing memiliki kolom kunci (biasanya kunci utama) untuk menghubungkan tabel tersebut dengan tabel lainnya. Ini berarti setiap tabel hasil partisi menyimpan subset dari kolom data namun tetap dapat dikaitkan dengan kolom kunci yang sama.

- **Contoh Partisi Vertikal**: Dalam sebuah tabel Karyawan, kolom-kolom yang terkait dengan data pribadi karyawan, seperti Nama, Tanggal_Lahir, dan Alamat, dapat disimpan dalam tabel terpisah Karyawan_Pribadi. Sementara itu, data yang berhubungan dengan pekerjaan, seperti Jabatan, Gaji, dan Tanggal_Bergabung, disimpan dalam tabel Karyawan_Pekerjaan. Kedua tabel ini memiliki kolom kunci KaryawanID yang sama, sehingga data karyawan dapat dihubungkan kembali melalui kunci tersebut.

Keuntungan Pemartisian Data

Pemartisian data memiliki beberapa manfaat utama yang membuatnya menjadi pilihan yang efektif untuk mengoptimalkan performa basis data:

- **Meningkatkan Performa Akses Data**: Dengan memecah tabel besar menjadi beberapa tabel kecil, query dapat diarahkan hanya pada bagian data yang relevan, sehingga mengurangi waktu pencarian dan akses data.
- **Pengelolaan Data Lebih Mudah**: Memecah data menjadi tabel-tabel lebih kecil memungkinkan tim basis data mengelola setiap bagian data secara lebih efisien dan mudah, terutama dalam hal backup, pemulihan, atau pemeliharaan data.
- **Optimalisasi Kapasitas Penyimpanan**: Pemartisian memungkinkan data yang lebih lama atau yang jarang diakses untuk disimpan pada perangkat penyimpanan yang lebih lambat, sedangkan data yang lebih sering diakses ditempatkan pada penyimpanan yang lebih cepat.
- **Skalabilitas**: Pemartisian data membuat basis data lebih mudah diskalakan, baik secara vertikal (menambah penyimpanan) maupun horizontal (menambah partisi), untuk menyesuaikan dengan volume data yang terus bertambah.

Tantangan dalam Pemartisian Data

Pemartisian data juga memiliki tantangan yang harus diperhatikan:

- **Kompleksitas Query**: Penggunaan pemartisian dapat membuat query menjadi lebih kompleks, karena data tersebar di beberapa tabel. Pengembang harus berhati-hati dalam merancang query agar dapat menarik data dari partisi yang berbeda.
- **Konsistensi Data**: Data yang dipartisi harus tetap konsisten, terutama jika data tersebut digunakan dalam berbagai proses atau aplikasi. Jika tidak dikelola dengan benar, pemartisian bisa menyebabkan inkonsistensi data.
- **Pemeliharaan Tambahan**: Pemartisian membutuhkan pemeliharaan tambahan, baik dalam hal backup maupun pengelolaan penyimpanan.

Normalisasi data

Normalisasi adalah proses penyusunan data dalam basis data untuk meminimalkan duplikasi atau redundansi dan memastikan integritas data. Dalam desain basis data, normalisasi dilakukan secara bertahap dengan mengikuti serangkaian aturan atau bentuk normal (normal forms) yang bertujuan untuk memperbaiki struktur tabel sehingga setiap tabel hanya menyimpan informasi yang relevan dengan entitasnya. Dengan cara ini, normalisasi membantu meningkatkan efisiensi basis data, mempermudah pemeliharaan data, dan menjaga agar basis data tetap terstruktur dengan baik.

Tujuan Normalisasi

Normalisasi bertujuan untuk:

1. **Menghilangkan Redundansi Data**: Normalisasi mengurangi data yang duplikat, sehingga data tersimpan lebih efisien dan mengurangi risiko inkonsistensi.
2. **Meningkatkan Integritas Data**: Dengan menghilangkan redundansi, normalisasi membantu menjaga keakuratan data, karena perubahan data hanya perlu dilakukan pada satu tempat.
3. **Mempermudah Pemeliharaan Data**: Struktur data yang terorganisir membuatnya lebih mudah untuk menambah, menghapus, atau memperbarui data tanpa memengaruhi keutuhan basis data secara keseluruhan.
4. **Mengoptimalkan Ruang Penyimpanan**: Penghapusan data yang berlebihan mengurangi kebutuhan penyimpanan, membuat penggunaan basis data lebih hemat.

Tahapan Normalisasi

Normalisasi biasanya diterapkan dalam beberapa tahapan atau bentuk normal (normal forms), di antaranya:

1. **Bentuk Normal Pertama (1NF)**
   - Pada tahap 1NF, setiap kolom dalam tabel harus menyimpan nilai tunggal, sehingga data berbentuk atomik (tidak terpisah-pisah). Ini berarti tidak boleh ada kolom yang menyimpan data yang terdiri dari beberapa nilai, seperti daftar nama atau alamat dalam satu kolom.
   - Contoh: Sebuah tabel Pesanan yang memiliki kolom Item tidak boleh menyimpan lebih dari satu item per baris, melainkan harus memecah setiap item ke baris terpisah.
2. **Bentuk Normal Kedua (2NF)**
   - Tahap 2NF memerlukan semua kolom non-kunci harus sepenuhnya bergantung pada kunci utama tabel. Jika ada kolom yang bergantung hanya pada sebagian dari kunci utama (pada tabel yang memiliki kunci gabungan), maka tabel tersebut belum memenuhi 2NF.
   - **Contoh**: Pada tabel Transaksi Penjualan yang memiliki kunci gabungan ID Transaksi dan Produk ID, kolom Harga Produk sebaiknya tidak ada di tabel ini jika harganya hanya bergantung pada Produk ID. Kolom ini sebaiknya dipindahkan ke tabel Produk untuk memenuhi aturan 2NF.
3. **Bentuk Normal Ketiga (3NF)**
   - Pada tahap 3NF, tabel tidak boleh memiliki dependensi transitif, di mana kolom non-kunci bergantung pada kolom non-kunci lain. Dengan kata lain, setiap kolom non-kunci harus bergantung langsung pada kunci utama.
   - Contoh: Dalam tabel Karyawan, kolom Kota sebaiknya tidak disimpan jika Kode_Pos sudah ada dan bergantung pada Kode Pos. Sebaliknya, Kota sebaiknya dipisahkan ke tabel lain dengan Kode_Pos sebagai kunci.

4. **Bentuk Normal Keempat (4NF) dan Bentuk Normal Kelima (5NF)**
   - Bentuk normal lebih tinggi ini mengatasi masalah lebih kompleks seperti independensi multi-nilai (4NF) dan pengelompokan independen (5NF). Kedua tahap ini biasanya dibutuhkan pada basis data yang lebih besar dan lebih kompleks, yang memiliki beberapa dependensi kompleks antar kolom.

Manfaat Normalisasi dalam Basis Data

1. **Mengurangi Duplikasi Data**: Dengan menghilangkan data berlebih, normalisasi mencegah penyimpanan data yang sama berulang kali di beberapa tabel.
2. **Mempercepat Pengelolaan dan Pembaruan Data**: Perubahan data menjadi lebih mudah, karena hanya perlu dilakukan di satu tempat. Misalnya, jika informasi kontak pelanggan berubah, perubahan hanya dilakukan pada satu tabel.
3. **Mengurangi Inkonsistensi Data**: Karena data tidak disimpan berulang kali, risiko terjadinya inkonsistensi data berkurang, sehingga data tetap akurat dan konsisten di seluruh sistem.
4. **Memudahkan Perancangan Query yang Efisien**: Struktur data yang teratur mempermudah pembuatan query yang lebih efektif dan cepat, serta memudahkan analisis data dalam basis data besar.

Tantangan Normalisasi

Meskipun normalisasi sangat bermanfaat, ada beberapa tantangan yang bisa muncul:

- **Kompleksitas Query**: Dalam beberapa kasus, data yang telah dinormalisasi tinggi membutuhkan penggabungan tabel yang lebih banyak saat mengambil data, yang dapat memperlambat akses.
- **Over-Normalisasi**: Dalam beberapa kasus, normalisasi berlebihan dapat membuat desain basis data terlalu kompleks untuk diimplementasikan dan diakses.
- **Kebutuhan Indeks Tambahan**: Normalisasi tinggi mungkin memerlukan pembuatan indeks pada beberapa kolom untuk meningkatkan kinerja query, yang dapat meningkatkan biaya penyimpanan dan pemeliharaan indeks.

Praktik Implementasi Normalisasi

1. **Identifikasi Kebutuhan Redundansi dan Efisiensi Data**: Sebelum memulai normalisasi, penting untuk memahami kebutuhan efisiensi dan reduksi data, apakah data tersebut memerlukan normalisasi tinggi atau cukup pada bentuk normal yang lebih rendah.
2. **Gunakan Bentuk Normal yang Sesuai**: Tidak semua basis data memerlukan 4NF atau 5NF. Untuk aplikasi sederhana, hingga 3NF biasanya sudah cukup untuk mengelola data secara optimal.
3. **Optimalkan dengan Denormalisasi**: Dalam beberapa kasus, denormalisasi atau pembalikan proses normalisasi diperlukan untuk meningkatkan performa query pada aplikasi yang sangat sering diakses.
4. **Penggunaan Indeks untuk Efisiensi Query**: Untuk basis data yang sangat besar, penambahan indeks pada kolom yang sering di-query bisa membantu mempercepat akses tanpa mengorbankan struktur normalisasi.

BAB 12
DESAIN LAPISAN KEAMANAN

12.1 MENDESAIN KEAMANAN PERANGKAT LUNAK

Desain keamanan perangkat lunak adalah proses menyusun mekanisme dan protokol untuk melindungi sistem komputer dari akses yang tidak sah, ancaman, dan potensi serangan yang bisa merusak integritas, ketersediaan, serta kerahasiaan data. Sistem keamanan umumnya bertujuan untuk memastikan bahwa hanya pengguna otoritatif yang dapat mengakses sistem, dan semua aktivitas dalam sistem terlindungi dari risiko yang bisa mengganggu layanan atau menimbulkan kerugian.

Kerentanan (Vulnerability) dalam Sistem

Kerentanan atau kelemahan dalam sistem komputer adalah celah atau kekurangan dalam perangkat keras atau perangkat lunak yang dapat dieksploitasi oleh pihak tak berwenang. Kerentanan ini bisa muncul akibat desain yang kurang aman, bug perangkat lunak, atau konfigurasi yang tidak tepat. Jika tidak diatasi, kerentanan bisa dimanfaatkan untuk mengakses data tanpa izin, mengubah informasi, atau bahkan merusak komponen sistem.

Contoh kerentanan meliputi:

- **Bug dalam Perangkat Lunak**: Kesalahan kode yang bisa dimanfaatkan untuk eksekusi kode tidak sah.
- **Konfigurasi yang Salah**: Setelan sistem yang tidak aman, seperti izin akses terlalu luas atau tidak adanya enkripsi.
- **Penggunaan Protokol yang Tidak Aman**: Protokol komunikasi yang tidak terlindungi atau sudah usang.

Ancaman dalam Sistem

Ancaman keamanan adalah potensi bahaya yang dapat mengganggu fungsi normal sistem atau mencuri data. Beberapa contoh ancaman meliputi:

1. Malware: Program jahat yang masuk ke dalam sistem untuk merusak atau mencuri informasi.
2. Phishing: Upaya memperoleh informasi sensitif seperti kata sandi dengan cara penipuan.
3. *Serangan DDoS (Distributed Denial of Service):* Serangan yang mencoba membanjiri sistem dengan lalu lintas berlebihan untuk menghentikan layanan.
4. Unauthorized Access: Akses oleh pengguna yang tidak diizinkan untuk melihat atau mengubah data tertentu.

12.2 MEKANISME KEAMANAN SISTEM

Untuk mengatasi kerentanan dan ancaman di atas, berbagai teknik dan mekanisme keamanan diterapkan:

1. **Firewall**

Firewall adalah penghalang pertama untuk melindungi sistem dari akses yang tidak sah. Firewall memfilter lalu lintas jaringan berdasarkan aturan keamanan, membatasi akses hanya untuk lalu lintas yang diizinkan.

Jenis Firewall: Ada beberapa jenis firewall, seperti firewall perangkat keras, firewall perangkat lunak, dan firewall berbasis jaringan, yang masing-masing berperan dalam mengamankan titik-titik akses berbeda.

2. **Proxy**

   Proxy bertindak sebagai perantara antara pengguna dan server tujuan. Proxy bisa menyembunyikan identitas pengguna, memfilter konten, dan menerapkan cache untuk mempercepat akses data yang sering diakses.

   Keamanan Melalui Proxy: Dengan menyembunyikan alamat IP asli dan memberikan lapisan keamanan tambahan, proxy dapat mengurangi risiko serangan langsung pada server utama.

3. **Secure Channel (Saluran Aman)**

   Secure channel mengacu pada saluran komunikasi yang terenkripsi, seperti HTTPS atau SSL/TLS, yang melindungi data dari pencurian atau manipulasi selama proses transmisi.

   Manfaat Enkripsi: Melindungi data sensitif selama transfer jaringan, terutama untuk transaksi yang membutuhkan privasi, seperti data keuangan atau informasi pribadi.

4. **Authentication (Otentikasi)**

   Otentikasi memastikan bahwa hanya pengguna yang sah yang dapat mengakses sistem. Teknik ini mengharuskan pengguna untuk memberikan kredensial yang valid, seperti kata sandi, biometrik, atau token.

Jenis Otentikasi:

1. **Single-Factor Authentication (SFA)**: Memerlukan satu langkah verifikasi, seperti kata sandi.
2. **Two-Factor Authentication (2FA)**: Menambahkan langkah kedua, seperti kode SMS atau email.
3. **Multi-Factor Authentication (MFA)**: Kombinasi beberapa faktor, seperti biometrik, kata sandi, dan token fisik untuk memperkuat keamanan.

Prinsip Dasar Desain Keamanan Sistem

Dalam mendesain keamanan perangkat lunak, beberapa prinsip utama diikuti untuk memastikan sistem aman dan tetap berfungsi dengan baik:

1. **Least Privilege**: Pengguna diberikan akses minimum yang mereka butuhkan untuk melakukan tugasnya. Ini mengurangi risiko penggunaan sistem yang berlebihan atau penyalahgunaan data.
2. **Defense in Depth**: Penerapan lapisan keamanan berlapis-lapis, seperti menggunakan firewall, otentikasi, dan enkripsi sekaligus. Jika satu lapisan berhasil ditembus, lapisan berikutnya tetap melindungi.
3. **Audit Trail dan Logging**: Mencatat semua aktivitas dalam sistem memungkinkan pengelola untuk mengidentifikasi tindakan mencurigakan dan melakukan investigasi atas pelanggaran keamanan.

4. **Encryption**: Data sensitif harus selalu dienkripsi, baik saat disimpan maupun saat ditransmisikan untuk menghindari pencurian data.
5. **Regular Security Updates**: Pembaruan perangkat lunak dan patch keamanan untuk memperbaiki bug dan celah yang diketahui dalam perangkat lunak.

12.3 PRAKTIK KEAMANAN DALAM PENGEMBANGAN PERANGKAT LUNAK

Dalam pengembangan perangkat lunak, beberapa langkah dapat diambil untuk meningkatkan keamanan:

1. **Code Review**: Melakukan pemeriksaan kode untuk mendeteksi potensi kerentanan dan memastikan bahwa kode telah mengikuti standar keamanan.
2. **Penetration Testing**: Pengujian simulasi serangan pada sistem untuk menemukan kerentanan sebelum dimanfaatkan oleh peretas.
3. **Secure Software Development Lifecycle (SDLC)**: Memasukkan prinsip-prinsip keamanan di setiap tahap pengembangan perangkat lunak, mulai dari perencanaan hingga pemeliharaan.
4. **User Education**: Melibatkan pengguna akhir dalam proses keamanan dengan memberikan edukasi tentang praktik keamanan dasar, seperti cara menjaga kata sandi dan mengenali upaya phishing.

Pemakai non otoritatif

Pemakai non otoritatif atau pengguna berbahaya (malicious user) adalah individu atau kelompok yang secara sengaja berusaha mengeksploitasi celah dalam sistem komputer atau jaringan untuk mendapatkan akses yang tidak sah, merusak data, mencuri informasi, atau bahkan menghentikan fungsi sistem secara menyeluruh. Malicious users dapat menggunakan berbagai metode dan teknik eksploitasi untuk menyerang kelemahan atau lubang keamanan dalam sistem, yang dapat berdampak serius bagi integritas, ketersediaan, dan kerahasiaan data.

Jenis Eksploitasi Oleh Malicious User

Metode yang sering digunakan oleh pengguna berbahaya dalam menyerang sistem antara lain:

1. **Malware (Malicious Software)**
   - ✓ **Virus**: Program yang menginfeksi file dan menggandakan diri untuk menyebar ke file lain, mengakibatkan kerusakan atau kehilangan data.
   - ✓ **Worm**: Berbeda dari virus, worm menyebar secara otomatis melalui jaringan tanpa perlu menumpang pada file lain, merusak sistem atau memenuhi jaringan sehingga menyebabkan kelambatan.
   - ✓ **Trojan Horse**: Program yang tampak sah tetapi berfungsi sebagai backdoor untuk mengakses sistem secara tersembunyi.
   - ✓ **Ransomware**: Malware yang mengenkripsi data dan meminta tebusan kepada pengguna untuk mendapatkan kembali akses ke data tersebut.
2. **Serangan Phishing dan Social Engineering**

- ✓ Teknik social engineering memanfaatkan manipulasi psikologis untuk memperoleh informasi sensitif pengguna, seperti kata sandi atau informasi pribadi.
- ✓ **Phishing**: Pesan palsu yang tampak resmi dikirim untuk menipu pengguna agar memberikan informasi sensitif, seperti data login atau nomor kartu kredit.

3. **SQL Injection**
   - ✓ Serangan ini dilakukan dengan memasukkan perintah SQL berbahaya ke dalam kolom input pada aplikasi, sehingga aplikasi mengeksekusi perintah tersebut pada basis data. SQL injection dapat menyebabkan kebocoran data, manipulasi, atau bahkan penghapusan data.
4. **Cross-Site Scripting (XSS)**
   - ✓ XSS memungkinkan penyerang memasukkan kode berbahaya ke dalam halaman web, yang kemudian dieksekusi di browser pengguna lain. Teknik ini sering digunakan untuk mencuri data atau melakukan serangan phishing.
5. **Denial of Service (DoS) dan Distributed Denial of Service (DDoS)**
   - ✓ Serangan DoS dilakukan dengan membanjiri sistem dengan lalu lintas berlebihan sehingga layanan menjadi lambat atau terhenti. DDoS adalah serangan DoS yang lebih kompleks dan melibatkan banyak sumber serangan secara simultan.
6. **Brute Force Attack**
   - ✓ Penyerang mencoba menebak kata sandi dengan mencoba semua kombinasi karakter. Meskipun metode ini membutuhkan waktu, pengguna yang menggunakan kata sandi lemah atau umum rentan terhadap serangan ini.
7. **Zero-Day Exploits**
   - ✓ Serangan ini memanfaatkan kelemahan yang belum diketahui oleh pengembang perangkat lunak atau belum ditambal. Karena belum ada perbaikan, serangan ini sangat efektif dan sulit dihentikan.

Kelemahan Sistem sebagai Titik Awal Serangan

Kelemahan dalam sistem, atau yang sering disebut sebagai vulnerability, merupakan titik awal yang dimanfaatkan oleh malicious users. Kelemahan ini bisa bersifat teknis, konfigurasi, atau kebijakan, dan dapat muncul karena:

1. **Kesalahan Kode**: Bug dalam kode perangkat lunak dapat menciptakan celah yang memungkinkan penyerang mengeksploitasi sistem.
2. **Konfigurasi yang Tidak Aman**: Konfigurasi standar atau kesalahan konfigurasi, seperti memberikan akses yang luas atau tidak menggunakan enkripsi, membuka peluang bagi penyerang.
3. **Kurangnya Enkripsi**: Data yang tidak dienkripsi saat transit atau penyimpanan membuat informasi tersebut rentan terhadap pencurian.
4. **Kebijakan Keamanan yang Tidak Ketat**: Tanpa kebijakan yang kuat, seperti autentikasi multifaktor, sistem mudah dieksploitasi oleh pengguna yang tidak sah.
5. **Keterlambatan Pembaruan**: Ketika patch keamanan tidak segera diterapkan, kerentanan yang dikenal menjadi celah bagi penyerang.

Dampak Eksploitasi oleh Pengguna Berbahaya

Eksploitasi oleh malicious users dapat merusak sistem secara menyeluruh. Dampak yang mungkin terjadi meliputi:

1. **Kehilangan Data**: Data bisa dicuri, dihapus, atau diubah sehingga memengaruhi integritas sistem.
2. **Gangguan Operasional**: Serangan DoS atau DDoS bisa menyebabkan sistem lambat atau tidak bisa diakses, menghentikan operasi bisnis.
3. **Kehilangan Kepercayaan Pengguna**: Kebocoran data pribadi atau data sensitif dapat merusak reputasi perusahaan dan kepercayaan pengguna.
4. **Kerugian Finansial**: Serangan siber sering kali menimbulkan biaya pemulihan yang tinggi, termasuk biaya hukum dan denda jika melibatkan data pribadi.

Kelemahan dalam sistem atau aplikasi, atau yang biasa disebut dengan *vulnerabilities*, adalah celah keamanan yang memungkinkan pihak tidak berwenang untuk mengakses atau merusak sistem. Berikut adalah beberapa kelemahan umum yang sering ditemui dan berpotensi dimanfaatkan oleh penyerang:

1. **Kata Kunci yang Lemah (Weak Password)**: Password yang mudah ditebak atau terlalu sederhana (misalnya, "password123" atau "admin") sangat rentan terhadap serangan. Kata kunci yang lemah membuka jalan bagi penyerang untuk tidak hanya mengakses satu komputer, tetapi juga berpotensi menyusup ke seluruh jaringan yang terhubung ke komputer tersebut. Ketika seorang penyerang berhasil menebak atau mencuri password, mereka dapat dengan mudah melakukan aksi berbahaya, seperti pencurian data, perusakan sistem, atau penyebaran malware.

   Langkah Pencegahan: Untuk mencegah kelemahan ini, pengguna disarankan untuk menerapkan kata kunci yang kuat dengan kombinasi huruf besar, huruf kecil, angka, dan karakter khusus, serta menggantinya secara berkala. Implementasi autentikasi dua faktor (2FA) atau multi-faktor juga dapat memperkuat keamanan.
2. **Perangkat Lunak yang Salah Konfigurasi (Misconfigured Software)**: Kesalahan dalam konfigurasi perangkat lunak adalah sumber umum dari celah keamanan dalam sistem. Konfigurasi yang salah dapat mencakup akses tidak terbatas, enkripsi yang tidak diaktifkan, atau pengaturan default yang belum diubah. Misalnya, jika akun pengguna lokal diatur dengan izin akses tinggi tanpa pembatasan, pengguna non-otoritatif atau malicious user dapat memanfaatkan kelemahan ini untuk menjalankan perintah berbahaya, mencuri informasi, atau menghancurkan data. Selain itu, kesalahan konfigurasi sering terjadi ketika perangkat lunak diinstal dan diatur tanpa mengacu pada pedoman keamanan.
   - **Contoh Kelemahan Konfigurasi**: Misalnya, memberikan hak administratif pada akun yang tidak memerlukan akses tersebut akan membuka risiko bagi penyerang. Hak akses tinggi memungkinkan seorang penyerang melakukan aksi berbahaya, seperti menghapus file, mengubah konfigurasi jaringan, atau mengakses data sensitif.

- **Dampak dari Salah Konfigurasi**: Jika tidak dikendalikan, kelemahan ini dapat berdampak besar pada integritas, ketersediaan, dan kerahasiaan data. Bahkan konfigurasi sederhana yang salah pada firewall atau server dapat menjadi pintu masuk bagi serangan yang lebih luas.
    - *Langkah Pencegahan:* Untuk mencegah kesalahan konfigurasi, tim keamanan IT harus menerapkan pengaturan keamanan yang ketat sesuai dengan panduan atau standar industri. Melakukan audit dan pemeriksaan keamanan secara berkala pada perangkat lunak dan perangkat keras dapat membantu menemukan dan memperbaiki kelemahan konfigurasi ini. Penggunaan prinsip *least privilege*, yaitu memberikan hak akses minimum yang dibutuhkan untuk menjalankan tugas, adalah langkah penting dalam mengurangi risiko.

3. **Rekayasa Sosial (Social Engineering)**: Rekayasa sosial adalah teknik manipulasi psikologis yang digunakan oleh penyerang untuk mendapatkan informasi sensitif dari pengguna. Dalam hal ini, penyerang memanfaatkan ketidaktahuan atau kepercayaan pengguna untuk mendapatkan akses kata sandi atau data pribadi lainnya. Misalnya, pengguna yang tidak menyadari bahwa kata sandinya harus dirahasiakan mungkin memberikannya kepada seseorang yang terlihat sah namun memiliki maksud buruk, seperti melalui telepon, email, atau pertemuan langsung. Serangan ini mengandalkan interaksi manusia dan dapat mencakup metode seperti *phishing* (email atau pesan palsu), *baiting* (menarik perhatian pengguna dengan iming-iming hadiah), atau *pretexting* (menggunakan identitas palsu).

   Upaya Pencegahan: Untuk mencegah serangan social engineering, pengguna harus diberi pelatihan tentang pentingnya menjaga kerahasiaan kata sandi dan cara mengenali tanda-tanda rekayasa sosial. Kebijakan keamanan perusahaan juga harus menegaskan bahwa informasi sensitif tidak boleh dibagikan tanpa verifikasi identitas pihak yang meminta.

4. **Koneksi Internet (Internet Connection)**: Banyak kerentanan keamanan dapat terjadi karena konfigurasi yang kurang aman pada koneksi internet. Sebagai contoh, server web yang diinstal dengan pengaturan standar seperti IIS (Internet Information Services) versi 5.0 pada Windows dapat membuka port atau layanan yang memungkinkan akses tidak sah. Selain itu, koneksi langsung melalui modem atau akses publik dapat melewati firewall, yang pada akhirnya membuka jaringan untuk disusupi dari luar. Koneksi internet yang tidak dikonfigurasi dengan tepat, seperti tidak menggunakan firewall atau pembatasan akses pada port, akan memberikan jalan masuk bagi penyerang.

   Upaya Pencegahan: Melakukan konfigurasi keamanan tambahan seperti menutup port yang tidak digunakan, menerapkan firewall untuk mengatur lalu lintas jaringan, dan menggunakan VPN untuk koneksi jarak jauh dapat mengurangi risiko. Selain itu, server harus diatur dengan pengaturan minimal yang diperlukan untuk menjalankan aplikasi.

5. **Transfer Data yang Tidak Terenkripsi**: Jika data ditransfer antara server dan pengguna dalam format teks biasa (unencrypted), informasi sensitif dapat dengan mudah ditangkap dan dimodifikasi oleh penyerang. Ketika data sensitif seperti kata sandi, nomor kartu kredit, atau informasi pribadi dikirim dalam bentuk yang tidak terenkripsi, peretas dapat menangkap data tersebut dengan teknik seperti *packet sniffing*, yang memungkinkan mereka untuk membaca dan memodifikasi data tersebut.
   Upaya Pencegahan: Agar data lebih aman, enkripsi harus diterapkan pada seluruh lalu lintas jaringan yang mengandung informasi sensitif. Penggunaan protokol seperti SSL/TLS untuk enkripsi data dalam transmisi akan menjaga integritas data. Selain itu, praktik penggunaan HTTPS untuk situs web sangat penting untuk menjamin keamanan data.
6. **Memori Penyangga yang Berlebih**: Kelemahan ini dikenal sebagai buffer overflow, yaitu situasi di mana sebuah aplikasi menerima data lebih banyak dari kapasitas memori penyangganya. Jika pemakai atau penyerang mengisi buffer secara berlebihan, data tersebut akan "meluap" ke memori sekitar buffer, yang menyebabkan aplikasi atau bahkan sistem operasi menjadi crash atau berhenti bekerja. Dalam beberapa kasus, buffer overflow ini memungkinkan penyerang menjalankan kode berbahaya pada sistem target, sehingga dapat mengambil alih kendali atau mendapatkan akses ilegal ke sumber daya sistem.
7. **Injeksi Perintah SQL (SQL Injection)**: Injeksi SQL adalah teknik eksploitasi yang digunakan oleh penyerang untuk memasukkan atau menyisipkan perintah SQL yang berbahaya ke dalam input aplikasi. Jika input tersebut tidak divalidasi dengan benar, perintah SQL yang disisipkan akan dijalankan oleh basis data dan dapat menyebabkan sejumlah efek negatif. Dengan menggunakan SQL injection, penyerang dapat mengakses, mengubah, atau bahkan menghapus data di dalam basis data yang seharusnya tidak dapat diakses. Injeksi SQL ini adalah salah satu bentuk serangan umum yang dapat merusak integritas dan kerahasiaan data.
8. **Kerahasiaan di Dalam Kode (Secret in Code)**: Menyimpan informasi sensitif seperti kata sandi, kunci enkripsi, atau data penting lainnya langsung di dalam kode program merupakan praktik yang sangat berisiko. Hal ini memungkinkan penyerang untuk menemukan dan mengeksploitasi informasi rahasia tersebut melalui metode seperti dekompilasi kode atau analisis binari. Rahasia yang disimpan di dalam kode juga mudah terekspos pada siapa pun yang memiliki akses terhadap kode sumber, yang dapat menimbulkan ancaman serius terhadap keamanan sistem.

Strategi Perancangan Keamanan Sistem :

1. Mempercayakan Konfigurasi Keamanan yang Telah Teruji

- **Penggunaan Solusi yang Teruji**: Dalam merancang sistem keamanan, sebaiknya menggunakan konfigurasi dan solusi yang telah terbukti efektif. Ini mengurangi risiko yang berasal dari kesalahan desain atau implementasi. Misalnya, menggunakan firewall atau perangkat lunak keamanan yang sudah terbukti aman dan memiliki track record yang baik.

- **Standarisasi**: Mengikuti standar industri atau praktik terbaik (best practices) dalam pengaturan konfigurasi keamanan. Misalnya, mengikuti standar ISO/IEC 27001 untuk sistem manajemen keamanan informasi.
- **Pembaruan dan Pemeliharaan**: Pastikan bahwa konfigurasi keamanan selalu diperbarui untuk menghadapi ancaman baru. Perangkat lunak dan sistem harus diperbarui secara berkala untuk menutup celah keamanan.

2. Validasi Masukan Eksternal

- **Pentingnya Validasi**: Semua masukan yang diterima dari pengguna atau sistem eksternal harus divalidasi. Ini bertujuan untuk mencegah serangan seperti SQL injection, cross-site scripting (XSS), dan buffer overflow yang sering memanfaatkan masukan yang tidak tervalidasi.
- **Teknik Validasi**: Gunakan berbagai teknik untuk memvalidasi masukan, seperti:
  - **Validasi Tipe Data**: Memastikan bahwa data yang diterima sesuai dengan tipe data yang diharapkan (misalnya, angka, string).
  - **Sanitisasi Data**: Menghapus karakter berbahaya atau tidak diinginkan dari masukan.
  - **Penggunaan Whitelist**: Membatasi masukan hanya pada nilai atau format tertentu yang dianggap aman.
- **Error Handling**: Pastikan sistem menangani kesalahan dengan baik tanpa memberikan informasi yang bisa dimanfaatkan oleh penyerang.

3. Mengasumsikan Sistem Eksternal Tidak Aman

- **Kewaspadaan Terhadap Data Sensitif**: Jika aplikasi menerima data sensitif dari sistem eksternal, seperti informasi pribadi atau finansial, perlu diasumsikan bahwa data tersebut bisa saja tidak aman. Oleh karena itu, harus ada mekanisme untuk mengenkripsi data ini selama transmisi dan penyimpanan.
- **Enkripsi**: Gunakan protokol enkripsi seperti TLS/SSL untuk melindungi data selama transmisi. Pastikan bahwa data sensitif yang disimpan dalam basis data juga dienkripsi untuk mengurangi risiko pencurian data.
- **Audit dan Monitoring**: Lakukan audit secara berkala dan monitoring terhadap interaksi dengan sistem eksternal. Ini penting untuk mendeteksi dan merespons potensi ancaman dengan cepat.

4. Menerapkan Kewenangan Akses yang Sangat Minimal

- **Prinsip Least Privilege (Kewenangan Terkecil)**: Prinsip ini menyatakan bahwa setiap pengguna, program, atau sistem harus memiliki akses minimum yang diperlukan untuk menjalankan fungsinya. Ini berarti bahwa pemakai biasa tidak seharusnya memiliki akses lebih dari yang mereka butuhkan untuk menyelesaikan tugas mereka. Hal ini membantu membatasi potensi kerusakan yang dapat ditimbulkan oleh kesalahan manusia atau serangan yang berhasil.
- **Pengelolaan Hak Akses**: Pengelolaan hak akses harus dilakukan dengan ketat. Ini termasuk:

- **Peninjauan Berkala**: Melakukan audit secara berkala terhadap hak akses pengguna untuk memastikan bahwa mereka masih sesuai dengan kebutuhan fungsional.
- **Penggunaan Role-Based Access Control (RBAC)**: Mengelompokkan pengguna ke dalam peran tertentu yang memiliki hak akses yang sama. Ini memudahkan pengelolaan dan pemantauan hak akses.

- **Akun Layanan**: Hindari mengaktifkan atribut yang berlebihan pada akun layanan (service accounts). Akun layanan sering memiliki hak akses tinggi, sehingga sangat penting untuk membatasi hak akses mereka hanya untuk fungsi yang benar-benar dibutuhkan.

5. Mengurangi Komponen dan Data yang Tersedia

- **Minimisasi Data dan Fungsi**: Setiap aplikasi atau sistem harus dirancang untuk menyediakan hanya fungsi yang diperlukan dan data yang relevan. Dengan mengurangi jumlah komponen yang tersedia, risiko terhadap serangan dapat diminimalkan.
- **Prinsip Kecil dan Sederhana (KISS)**: Mengembangkan aplikasi dengan filosofi KISS (Keep It Simple, Stupid). Dengan menjaga desain sistem sesederhana mungkin, akan lebih mudah untuk mengidentifikasi dan memperbaiki potensi celah keamanan.
- **Penghapusan Fitur yang Tidak Diperlukan**: Secara aktif mengevaluasi fitur-fitur dalam aplikasi dan menghapus atau menonaktifkan fungsi-fungsi yang tidak digunakan. Setiap fitur tambahan dapat membuka potensi titik serangan baru.

6. Menggunakan Mode Keamanan Standar

- **Pengaturan Default yang Aman**: Selalu gunakan pengaturan default yang aman untuk perangkat lunak dan layanan yang digunakan. Jangan mengaktifkan fitur atau layanan yang tidak diperlukan, karena ini dapat menjadi celah bagi penyerang.
- **Penerapan Keamanan Berlapis**: Menggunakan pendekatan keamanan berlapis (defense in depth), dengan menonaktifkan layanan yang tidak diperlukan di setiap lapisan sistem. Ini mengurangi vektor serangan yang mungkin dimanfaatkan.
- **Dokumentasi dan Kebijakan**: Memiliki dokumentasi dan kebijakan yang jelas tentang apa yang diizinkan dan tidak diizinkan dalam penggunaan sistem. Ini mencakup panduan tentang bagaimana dan kapan mengaktifkan layanan tertentu.

7. Menerapkan Enkripsi Data dengan Algoritma yang Sulit Dipecahkan

- **Enkripsi Data Sensitif**: Semua data sensitif, baik saat transit maupun saat disimpan, harus dienkripsi untuk melindungi dari akses yang tidak sah. Ini termasuk informasi pribadi, finansial, dan informasi bisnis penting.
- **Pemilihan Algoritma Enkripsi**: Gunakan algoritma enkripsi yang diakui dan telah terbukti aman, seperti AES (Advanced Encryption Standard) dengan panjang kunci yang memadai (misalnya, 256 bit). Algoritma ini sulit dipecahkan dengan metode brute force dan tetap aman selama beberapa dekade.
- **Pengelolaan Kunci Enkripsi**: Keamanan enkripsi tidak hanya bergantung pada algoritma yang digunakan, tetapi juga pada cara kunci enkripsi dikelola. Kunci harus

disimpan dengan aman, tidak boleh disimpan bersamaan dengan data yang dienkripsi, dan harus dirotasi secara berkala.

12.4 MERENCANAKAN KEAMANAN APLIKASI

1. Memahami Ancaman yang Mungkin Terjadi pada Aplikasi

Sebelum merancang sistem keamanan, penting untuk memahami berbagai ancaman yang dapat menyerang aplikasi. Ini mencakup:

- **Serangan Jaringan**: Ancaman seperti *Distributed Denial of Service* (DDoS), man-in-the-middle, dan sniffing yang dapat mengganggu ketersediaan atau integritas aplikasi.
- **Serangan Aplikasi**: Ini mencakup serangan seperti SQL injection, *cross-site scripting* (XSS), dan command injection, yang dapat mengeksploitasi kerentanan dalam aplikasi untuk mencuri data, mengubah data, atau mengambil alih kendali aplikasi.
- **Ancaman Internal**: Ancaman ini berasal dari dalam organisasi, seperti karyawan yang memiliki niat buruk atau yang tidak sengaja melakukan kesalahan yang dapat merugikan keamanan aplikasi.
- **Malware**: Penggunaan perangkat lunak berbahaya yang dapat merusak data, mencuri informasi, atau merusak sistem.
- **Phishing**: Usaha untuk mendapatkan informasi sensitif dengan menyamar sebagai entitas yang terpercaya dalam komunikasi elektronik.

2. Mengidentifikasi Ancaman-ancaman Aplikasi (Threat Modelling)

Threat modeling adalah proses sistematis untuk mengidentifikasi, memahami, dan mengelola risiko keamanan yang terkait dengan aplikasi. Langkah-langkah dalam threat modeling meliputi:

- **Pengumpulan Informasi**: Memahami arsitektur aplikasi, komponen, dan interaksinya. Ini termasuk pemetaan alur data dan titik masuk ke aplikasi.
- **Identifikasi Ancaman**: Menggunakan framework dan teknik yang ada untuk mengidentifikasi ancaman potensial. Beberapa metode yang umum digunakan adalah:
  - **STRIDE**: Model ini mengidentifikasi enam kategori ancaman: Spoofing (penipuan identitas), Tampering (perubahan data), Repudiation (penyangkalan), Information Disclosure (pengungkapan informasi), Denial of Service (denial of service), dan Elevation of Privilege (peningkatan hak akses).
  - **PASTA**: Model ini berfokus pada analisis risiko dan dampak dari serangan yang mungkin terjadi pada aplikasi.
- **Penilaian Risiko**: Menghitung risiko yang terkait dengan setiap ancaman berdasarkan kemungkinan terjadinya dan dampaknya terhadap aplikasi. Ini membantu dalam memprioritaskan ancaman mana yang harus ditangani terlebih dahulu.
- **Dokumentasi dan Komunikasi**: Mencatat hasil threat modeling dan berbagi dengan tim pengembangan dan pemangku kepentingan lainnya agar semua pihak memahami risiko yang ada.

3. Menetapkan Kebijakan Keamanan sebagai Tindakan Prefentif

Setelah mengidentifikasi ancaman, langkah selanjutnya adalah menetapkan kebijakan keamanan yang berfungsi sebagai tindakan prefentif. Kebijakan ini mencakup:

- **Kebijakan Akses**: Menetapkan siapa yang dapat mengakses aplikasi dan data, serta menetapkan level akses yang sesuai. Ini melibatkan penerapan prinsip Least Privilege (kewenangan terkecil) dan Role-Based Access Control (RBAC).
- **Prosedur Validasi Data**: Membangun mekanisme untuk memvalidasi semua masukan dari pengguna dan sistem eksternal. Pastikan bahwa semua data yang masuk telah disaring dan dienkripsi jika diperlukan.
- **Penggunaan Enkripsi**: Menerapkan enkripsi untuk melindungi data sensitif baik dalam transit maupun saat disimpan. Kebijakan ini juga harus mencakup pengelolaan kunci enkripsi dengan aman.

Mencegah Kegagalan Keamanan:

Kegagalan dalam merencanakan kebijakan keamanan sejak awal dapat mengakibatkan celah keamanan yang signifikan. Jika keamanan hanya dipikirkan setelah aplikasi selesai, ada risiko bahwa banyak masalah yang tidak terlihat akan muncul, yang sulit atau bahkan tidak mungkin untuk diperbaiki tanpa mengubah arsitektur dasar aplikasi.

Merencanakan dan Menerapkan Fitur Keamanan Selama Pengembangan

1. **Integrasi Keamanan dalam Proses Pengembangan:** Keamanan harus menjadi bagian dari setiap tahap pengembangan, mulai dari perencanaan hingga pengujian dan pemeliharaan. Ini termasuk penerapan prinsip-prinsip pengembangan perangkat lunak yang aman (Secure Software Development Life Cycle - SSDLC).
2. **Tahap-tahap Kebijakan Keamanan dalam Model MSF:** Model Microsoft Solutions Framework (MSF) memberikan kerangka kerja yang jelas untuk perencanaan dan penerapan kebijakan keamanan dalam pengembangan aplikasi. Tahapan-tahapan kebijakan keamanan dalam model MSF mencakup:
    1. **Inisiasi Proyek**:
        - **Identifikasi Stakeholder**: Melibatkan semua pemangku kepentingan, termasuk tim pengembang, manajemen, dan tim keamanan, untuk memahami kebutuhan keamanan aplikasi.
        - **Penentuan Tujuan Keamanan**: Menetapkan tujuan dan sasaran keamanan yang ingin dicapai, seperti perlindungan data pengguna dan kepatuhan terhadap regulasi.
    2. **Perencanaan Keamanan**:
        - **Pengembangan Kebijakan Keamanan**: Membuat dokumen kebijakan yang merinci aturan dan prosedur keamanan, termasuk pengelolaan akses, penggunaan enkripsi, dan penanganan insiden.
        - **Analisis Ancaman dan Risiko**: Melakukan analisis risiko untuk mengidentifikasi potensi ancaman dan mengembangkan strategi mitigasi.
    3. **Desain dan Pengembangan**:

- **Integrasi Keamanan dalam Desain Arsitektur**: Memastikan bahwa arsitektur aplikasi dirancang dengan mempertimbangkan keamanan, termasuk kontrol akses, enkripsi, dan perlindungan terhadap serangan umum.
- **Implementasi Kontrol Keamanan**: Menerapkan kontrol keamanan yang sesuai selama proses pengembangan, seperti validasi input, sanitasi data, dan pengelolaan sesi.

4. **Pengujian dan Validasi**:
   - **Uji Keamanan**: Melakukan pengujian keamanan untuk memastikan bahwa semua kontrol keamanan berfungsi dengan baik dan tidak ada kerentanan yang tersisa.
   - **Uji Kepatuhan**: Memastikan bahwa aplikasi memenuhi semua kebijakan dan regulasi yang relevan.
5. **Penerapan dan Pemeliharaan**:
   - **Peluncuran Aplikasi**: Setelah semua pengujian selesai, aplikasi diluncurkan dengan dokumentasi yang jelas mengenai kebijakan keamanan yang diterapkan.
   - **Monitoring dan Tanggapan Insiden**: Menerapkan sistem pemantauan untuk mendeteksi dan merespons insiden keamanan secara real-time.
6. **Evaluasi dan Perbaikan**:
   - **Peninjauan Berkala**: Melakukan peninjauan berkala terhadap kebijakan dan praktik keamanan untuk menyesuaikan dengan ancaman baru dan perubahan dalam lingkungan teknologi.

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 pada tahap planning.
Stabilizing	Menjalankan uji kemanan. Melakukan revisi terhadap model ancaman jika ditemukan informasi baru yang belum teridentifikasi

	selama pengembangan, serta dilakukan pengujian dan umpan balik dari pengguna.
Deploying	Monitoring ancaman-ancaman yang mungkin terjadi pada aplikasi.

Model Stride

adalah salah satu teknik yang sangat berguna dalam pengembangan perangkat lunak untuk mengidentifikasi dan mengkategorikan ancaman keamanan. Model ini membantu tim pengembang dan analis keamanan untuk secara sistematis mengidentifikasi potensi risiko yang mungkin dihadapi oleh aplikasi selama siklus hidupnya. STRIDE merupakan akronim dari enam jenis ancaman yang berbeda, yaitu: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, dan Elevation of Privilege. Berikut adalah penjelasan rinci dan pengembangan dari setiap kategori ancaman dalam model STRIDE:

1. **Spoofing Identity (Penipuan Identitas)**

   Spoofing adalah tindakan penyerang yang menyamar sebagai entitas lain dengan tujuan untuk mendapatkan akses atau informasi yang tidak sah. Dalam praktiknya, spoofing bisa melibatkan penipuan identitas pengguna, server, atau bahkan perangkat. Misalnya, seorang penyerang dapat mencuri kredensial pengguna melalui serangan phishing, lalu menggunakan informasi yang didapatkan untuk mengakses akun pengguna secara ilegal. Dengan menyamarkan identitas atau asal usul informasi, penyerang berusaha mengelabui sistem atau individu untuk memperoleh hak akses yang seharusnya tidak mereka miliki.

2. **Tampering (Pengubahan Data)**

   Tampering merujuk pada tindakan modifikasi data secara tidak sah, baik selama proses transmisi data maupun saat data disimpan di server. Misalnya, seorang penyerang dapat berhasil mengubah informasi yang ada dalam basis data, seperti memanipulasi jumlah uang dalam sebuah transaksi keuangan, dengan tujuan untuk meraih keuntungan pribadi atau merusak integritas sistem.

3. **Repudiation (Penyangkal)**

   Repudiation terjadi ketika seorang pengguna menyangkal bahwa mereka melakukan suatu tindakan, seperti melakukan transaksi atau mengakses data. Ini bisa menimbulkan masalah dalam hal pertanggungjawaban.Misalnya Seorang pengguna yang melakukan pembelian online dapat menolak melakukan transaksi jika tidak ada bukti yang jelas bahwa mereka melakukannya.

4. **Information Disclosure (Pengungkapan Informasi)**

   Information disclosure terjadi ketika informasi sensitif terungkap kepada pihak yang tidak berwenang. Ini dapat terjadi secara tidak sengaja atau sebagai hasil dari serangan. Misalnya Data pengguna seperti nama, alamat, dan informasi keuangan bisa bocor akibat serangan SQL injection atau kesalahan konfigurasi keamanan.

5. **Denial of Service (Penolakan Layanan)**

Denial of Service (DoS) adalah serangan yang bertujuan untuk membuat layanan tidak tersedia bagi pengguna yang sah dengan membanjiri sistem dengan permintaan yang berlebihan. Misalnya Penyerang dapat melakukan serangan DDoS (Distributed Denial of Service) dengan menggunakan botnet untuk mengirimkan lalu lintas berlebih ke server, membuatnya tidak dapat diakses.

6. **Elevation of Privilege (Peningkatan Hak Akses)**
   Elevation of Privilege terjadi ketika seorang pengguna biasa mendapatkan hak akses yang lebih tinggi dari yang seharusnya, memungkinkan mereka melakukan tindakan yang seharusnya tidak dapat mereka lakukan. Misalnya Seorang pengguna biasa dapat mengeksploitasi kerentanan dalam aplikasi untuk mendapatkan akses admin dan melakukan perubahan yang merugikan.

Setelah ancaman-ancaman terhadap keamanan sistem teridentifikasi, langkah selanjutnya adalah merespons dan menangani ancaman tersebut dengan cara yang efektif melalui serangkaian langkah terstruktur. Pertama, lakukan analisis risiko untuk setiap ancaman yang diidentifikasi dan prioritaskan berdasarkan tingkat risikonya. Kemudian, kembangkan rencana tindakan yang jelas untuk setiap ancaman, termasuk pendelegasian tanggung jawab kepada anggota tim. Implementasikan kontrol keamanan yang sesuai, seperti autentikasi multifaktor untuk mengatasi spoofing, enkripsi untuk melindungi integritas data dari tampering, dan perangkat lunak pencegah serangan DoS.

Pemantauan aktif menggunakan sistem pemantauan dan logging diperlukan untuk mendeteksi aktivitas mencurigakan. Siapkan rencana tanggap darurat untuk merespons insiden keamanan, termasuk komunikasi yang jelas kepada semua pihak yang terlibat. Setelah insiden diatasi, lakukan pemulihan sistem dan analisis pasca insiden untuk mengidentifikasi kelemahan serta memperbarui kebijakan dan prosedur keamanan. Selanjutnya, berikan pelatihan berkala kepada karyawan tentang ancaman dan praktik terbaik keamanan, dan terus evaluasi dan tingkatkan rencana serta prosedur keamanan agar tetap relevan dan efektif terhadap ancaman yang berkembang. Dengan pendekatan ini, organisasi dapat lebih siap menghadapi dan mengatasi ancaman, sekaligus meningkatkan keamanan sistem secara keseluruhan.Merespon ancaman-ancaman keamanan aplikasi adalah langkah penting untuk melindungi pengguna dan sistem. Terdapat beberapa cara yang dapat dilakukan untuk menangani ancaman tersebut:

1. **Menginfokan kepada Pengguna**: Penting untuk menginformasikan pengguna tentang potensi ancaman yang mungkin mereka hadapi saat menggunakan aplikasi. Ini bisa dilakukan melalui pemberitahuan, dokumentasi, atau pembaruan perangkat lunak yang menjelaskan risiko keamanan dan cara-cara untuk melindungi diri mereka, seperti pengaturan kata sandi yang kuat dan penghindaran dari praktik phishing.
2. **Memberikan Peringatan tentang Kelemahan Sistem**: Ketika kelemahan terdeteksi selama aktivitas pengguna, aplikasi harus dapat memberikan peringatan. Misalnya, jika pengguna mencoba melakukan tindakan tertentu yang berpotensi menimbulkan risiko keamanan, sistem harus menginformasikan bahwa aktivitas tersebut dapat

membahayakan data atau integritas aplikasi. Ini dapat membantu pengguna membuat keputusan yang lebih baik tentang bagaimana melanjutkan.

3. **Menghapus Fitur Berisiko**: Jika ada fitur tertentu dalam aplikasi yang diketahui dapat menyebabkan risiko keamanan signifikan dan tidak dapat dikurangi secara efektif, penting untuk mempertimbangkan untuk menghapus fitur tersebut dari aplikasi. Keputusan ini harus didasarkan pada analisis risiko yang mendalam, serta umpan balik dari pengguna untuk memastikan bahwa keamanan menjadi prioritas utama tanpa mengorbankan fungsionalitas penting.
4. **Mengidentifikasi Teknik Meringankan (Mitigasi) Risiko**: Jika pengguna memilih untuk menambahkan fitur baru, penting untuk menetapkan teknik mitigasi risiko yang relevan. Beberapa teknik mitigasi risiko yang dapat dipertimbangkan meliputi:
    - **Validasi Input**: Memastikan bahwa semua data yang dimasukkan oleh pengguna divalidasi untuk mencegah serangan seperti SQL injection dan Cross-Site Scripting (XSS).
    - **Enkripsi Data**: Menggunakan enkripsi untuk melindungi data sensitif baik dalam transit maupun saat disimpan, sehingga mengurangi risiko kebocoran informasi.
    - **Audit dan Logging**: Mengimplementasikan mekanisme audit dan logging untuk melacak aktivitas pengguna, memungkinkan deteksi dini terhadap perilaku mencurigakan dan memberikan informasi untuk analisis pasca-insiden.
    - **Pengujian Keamanan**: Secara teratur melakukan pengujian keamanan pada aplikasi untuk mengidentifikasi dan mengatasi kerentanan sebelum dapat dieksploitasi oleh penyerang.

1. Autentikasi dan Otorisasi (Authentication and Authorization)

Autentikasi adalah proses verifikasi identitas seorang pengguna atau entitas yang ingin mengakses sistem. Ini biasanya dilakukan melalui kombinasi nama pengguna dan kata sandi, tetapi juga bisa melibatkan metode yang lebih kuat seperti autentikasi multifaktor (MFA), yang memerlukan pengguna untuk memberikan beberapa bukti identitas sebelum diizinkan masuk.

Otorisasi, di sisi lain, adalah proses yang mengikuti autentikasi dan menentukan hak akses yang diberikan kepada pengguna yang telah terautentikasi. Ini memastikan bahwa pengguna hanya dapat mengakses sumber daya dan melakukan tindakan yang sesuai dengan peran dan tanggung jawab mereka. Misalnya, seorang pengguna dengan peran "admin" mungkin memiliki akses penuh untuk mengelola data, sementara pengguna dengan peran "pengguna biasa" hanya memiliki akses terbatas untuk melihat data.

2. Komunikasi yang Aman (Secure Communication)

Ketika data ditransmisikan melalui jaringan atau dimuat dalam antrian untuk dieksekusi, penting untuk memastikan bahwa komunikasi tersebut aman. Ini melibatkan penggunaan protokol keamanan seperti Secure Socket Layer (SSL) atau Transport Layer Security (TLS), yang mengenkripsi data dalam perjalanan antara klien dan server. Dengan enkripsi, informasi sensitif seperti kredensial login atau data pribadi tidak dapat dibaca oleh pihak yang tidak berwenang yang mungkin mencegat komunikasi tersebut. Selain itu, teknik

seperti VPN (Virtual Private Network) dapat digunakan untuk menciptakan saluran komunikasi yang aman di antara jaringan yang berbeda.

3. Kualitas Layanan (Quality of Service)

Kualitas Layanan merujuk pada pengelolaan dan pemantauan layanan untuk memastikan bahwa sistem dapat memenuhi ekspektasi pengguna dalam hal ketersediaan, keandalan, dan kecepatan. Dalam konteks keamanan, implementasi kualitas layanan dapat mencakup pembuatan profil untuk pesan yang dikirim ke dalam sistem, memastikan bahwa pesan yang lebih kritis mendapatkan prioritas tinggi, dan mengurangi risiko kemacetan atau kegagalan sistem.

4. Membatasi atau Menghambat (Throttling)

Throttling adalah teknik yang digunakan untuk membatasi jumlah permintaan yang dapat dikirim ke sistem dalam jangka waktu tertentu. Ini sangat penting untuk melindungi aplikasi dari serangan DDoS (Distributed Denial of Service), di mana penyerang mencoba membanjiri sistem dengan lalu lintas berlebih. Dengan menerapkan throttling, sistem dapat menjaga ketersediaan layanan bagi pengguna yang sah dan mengurangi kemungkinan terjadinya kelebihan beban.

5. Pemeriksaan (Auditing)

Pemeriksaan adalah proses pengumpulan informasi tentang aktivitas pengguna dan kejadian penting dalam sistem. Informasi ini biasanya disimpan dalam log audit yang dapat dianalisis untuk mendeteksi perilaku mencurigakan atau pelanggaran kebijakan keamanan. Melalui audit yang rutin, organisasi dapat menilai kepatuhan terhadap kebijakan keamanan, mengidentifikasi potensi kerentanan, dan melakukan perbaikan yang diperlukan.

6. Penyaringan (Filtering)

Penyaringan adalah proses yang melibatkan intersepsi dan pengujian pesan yang dikirim ke dalam sistem untuk memastikan bahwa hanya data yang aman dan valid yang diterima. Ini dapat dilakukan dengan menggunakan firewall, sistem deteksi intrusi (IDS), dan perangkat lunak antivirus yang secara aktif memeriksa data untuk mendeteksi dan mencegah ancaman. Dengan cara ini, sistem dapat melindungi diri dari malware, virus, dan serangan lainnya yang dapat merusak integritas dan keamanan data.

7. Pembatasan Hak Akses

Pembatasan hak akses adalah prinsip dasar dalam keamanan informasi yang menetapkan bahwa pengguna hanya diberikan akses yang minimal untuk menyelesaikan tugasnya. Ini membantu meminimalkan risiko penyalahgunaan akses, di mana pengguna dengan hak akses berlebih dapat merusak data atau sistem. Mengimplementasikan model akses berbasis peran (RBAC) adalah cara yang efektif untuk menerapkan pembatasan hak akses, memastikan bahwa pengguna hanya dapat melakukan tindakan yang diperlukan berdasarkan peran mereka.

Mitigation techniquetigat	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-resistants protocols	T,R
Time stamps	R
Filtering	D
Throttling	D
Quality of service	D
Run with least privilege	E

BAB 13
POLA PIKIR KONSTRUKSI DIGITAL

"Masa depan sudah ada di sini, hanya saja penyebarannya tidak merata."
– William Gibson

Prediksi tentang masa depan teknologi cenderung tidak pasti. Orang suka mengatakan sesuatu akan "mengganggu," atau bahwa teknologi itu "mati." Tidak ada yang benar-benar berfungsi seperti itu. Masa depan teknologi konstruksi akan lebih atau kurang menakjubkan daripada yang dapat diprediksi siapa pun, karena tidak seorang pun dapat memprediksi apa yang akan ditemukan, efek mengejutkan apa di dunia yang akan menghasilkan kemungkinan baru. Kita juga tidak dapat memprediksi teknologi brilian apa yang akan gagal karena diluncurkan dengan buruk atau didanai dengan buruk.

Industri konstruksi saat ini sangat kompleks, dan terdiri dari banyak segmen. Ini tidak akan berubah di masa depan, meskipun saya pikir akan ada beberapa tingkat konsolidasi. Dalam bab terakhir ini, kita berkesempatan untuk meninjau kembali pola pikir yang dibahas dalam Bab 1, dan menggunakannya sebagai lensa untuk melihat perubahan yang mungkin terjadi dalam industri, dan bagaimana hal itu akan memengaruhi perusahaan besar dan canggih, serta perusahaan kecil yang lebih fokus.

13.1 MENGADOPSI POLA PIKIR KONSTRUKSI DIGITAL

Teknologi meningkatkan kemampuan manusia, karena kita memisahkan keputusan intuitif dari keputusan digital, dengan masing-masing saling mendukung tergantung pada konteksnya. Mari kita lihat bagaimana masing-masing akan berkembang seiring waktu.

Arena Keterampilan Manusia yang Intuitif

Pada tahun 2035, masih akan terjadi kekurangan keterampilan yang akut. Hal ini akan didorong oleh permintaan, demografi, dan perubahan teknologi:

Permintaan: Dunia akan terus membutuhkan konstruksi dalam jumlah besar, baik untuk menampung populasi yang akan terus bertambah, terutama di Afrika, tetapi juga untuk merenovasi dan mendekonstruksi bangunan saat kita secara radikal memikirkan kembali bagaimana bangunan berinteraksi dengan lingkungan di seluruh siklus hidupnya. Secara khusus, kemungkinan besar konsumsi energi di lingkungan binaan akan menjadi fokus regulasi dan insentif lainnya, yang menyebabkan banyak perusahaan konstruksi skala kecil dan menengah menjual layanan yang menguntungkan untuk merenovasi bangunan tua. Perombakan dan pembongkaran merupakan jenis proyek yang sangat tidak teratur, dan karenanya akan memerlukan lebih banyak pengawasan manusia daripada proyek lain dengan ukuran dan kompleksitas yang sama.

Perubahan iklim akan membahayakan ratusan kota dan kota pesisir, yang mengarah pada proyek konstruksi yang berkisar dari pembangunan tembok dan tanggul hingga perombakan lebih lanjut bangunan agar tahan banjir, hingga pembongkaran dan pembangunan kembali – pembongkaran itu sendiri akan tunduk pada peraturan baru.

Banyak negara lain, ada juga kebutuhan mendasar untuk penggantian dan peningkatan infrastruktur yang tidak akan segera selesai; bahkan, kemungkinan akan menjadi tingkat aktivitas yang terus meningkat, atau investasi berkala.

Demografi: Profil populasi negara didorong oleh kelahiran, diukur sebagai kelahiran hidup per wanita. Tingkat penggantian untuk ukuran ini, jika populasi suatu negara tetap stabil, adalah 2,1 kelahiran hidup per wanita. Hampir tidak ada negara maju yang memiliki ukuran ini di atas 2,0 saat ini, dan diperkirakan akan terus menurun atau tetap pada tingkat saat ini tanpa batas. Tidak akan ada cukup orang untuk mempertahankan tingkat aktivitas ekonomi saat ini pada tahun 2035, setidaknya tidak saat kita memproduksi tingkat tersebut saat ini.

Perubahan teknologi: Dalam ekonomi secara keseluruhan, keterampilan akan menjadi usang dalam waktu sekitar 5–10 tahun. Hal ini karena perubahan teknologi di seluruh industri terus mengarah pada peningkatan produktivitas, keselamatan, dan kualitas; perubahan tidak pernah berhenti. Selalu ada alat atau keterampilan baru yang, setelah diperkenalkan, segera diharapkan dari semua orang.

Konstruksi sebagian besar kebal terhadap hal ini, meskipun pada tahun 2020 hal itu sudah mulai berubah, karena lebih banyak alat digital ditambahkan ke perangkat. Intinya di sini adalah, seiring dengan perubahan keterampilan yang dibutuhkan, akan selalu ada jeda antara berapa banyak pekerja terampil yang dibutuhkan dan berapa banyak pekerja yang memiliki keterampilan itu.

13.2 TEKNOLOGI PEMBELAJARAN

Karena akan terus terjadi kekurangan keterampilan secara keseluruhan, kita dapat memperkirakan bahwa pembelajaran akan menjadi kompetensi inti perusahaan konstruksi, baik itu kontraktor umum, atau perdagangan khusus, atau perusahaan lainnya. Perubahan ini telah mengakar dalam barang-barang konsumen, perangkat lunak, dan industri lainnya, karena alasan yang sama – mereka tidak memiliki cukup pekerja yang sangat terampil, dan perguruan tinggi merupakan penghasil banyak keterampilan yang buruk – jadi perusahaan-perusahaan ini telah menjadi sangat canggih dalam melatih sejumlah besar staf. Dan melakukan pelatihan itu secara berkelanjutan karena keterampilan terus berkembang.

Teknologi pembelajaran sering dikelompokkan menjadi dua kategori – pelatihan dan dukungan kinerja. Pelatihan dapat berupa apa saja mulai dari simulasi hingga kelas hingga video daring. Dugaan saya adalah sebagian besar pelatihan di tahun-tahun mendatang akan dilakukan dalam realitas virtual, atau apa pun yang menggantikannya – teknologinya sudah sangat bagus, dan akan terus meningkat seiring dengan kemajuan kecerdasan buatan dan teknologi pendukung lainnya. Simulasi secara khusus akan disesuaikan untuk mengajarkan operator manusia jenis keterampilan yang sangat bergantung pada konteks yang hanya dapat dilakukan oleh manusia, dengan menempatkan mereka dalam banyak skenario, dengan cepat. Pelatihan semacam ini sekarang digunakan untuk manajemen ritel dan pekerjaan lain yang sangat menghargai kemampuan mengenali pola dan merespons dengan cepat. Kita dapat berharap bahwa simulasi semacam ini akan dikembangkan dalam perdagangan konstruksi dan GC, sehingga anggota tim baru dapat dilatih lebih cepat daripada metode tradisional saja.

Proses magang menjadi pekerja tetap masih diperlukan, tetapi dapat ditingkatkan secara signifikan melalui pelatihan simulasi.

Dukungan kinerja melibatkan penyediaan informasi tepat di mana dan kapan dibutuhkan. Saat ini, hal ini sering kali sulit dilakukan, karena sulit untuk mengetahui apa yang diperlukan, kapan. Namun, seiring dengan semakin matangnya realitas tertambah dan teknologi lainnya, kita dapat berharap bahwa agen AI atau yang serupa akan dapat memberikan informasi terperinci tepat pada saat dibutuhkan – bahkan kemungkinan besar akan menjadi informasi yang cukup canggih. Kedua cara belajar ini akan mengembangkan keterampilan manusia yang dibutuhkan di masa depan, dan pembaca akan mampu membayangkan sebagian besar konten yang akan dipelajari. Namun, di luar topik dan keterampilan standar untuk perdagangan tertentu, akan ada berbagai keterampilan baru yang belum kita kembangkan – keterampilan yang tumbuh dari analisis data. Karena perangkat lunak dan teknologi konstruksi digital terus menjadi lebih canggih, dan saat kita mengumpulkan informasi yang lebih banyak dan lebih baik dari lokasi kerja, rantai pasokan, dan dunia yang lebih luas, kemampuan untuk mengelola data tersebut, mengatur analisis yang berguna, dan menindaklanjuti analisis tersebut akan menjadi sangat penting. Singkatnya, masa depan konstruksi akan menjadi masa depan pemberdayaan dan pembelajaran yang terorganisasi, di mana pekerja lapangan diberikan pelatihan dan kinerja di tempat untuk melakukan pekerjaan mereka dengan aman dan pada tingkat produktivitas yang tampaknya seperti keajaiban saat ini – kita tahu ini karena lintasan yang sama telah terjadi di industri lain.

13.3 KETERAMPILAN DIGITAL

Bagian kedua dari pola pikir baru kita berkaitan dengan pemikiran konstruksi dalam konteks digital. Konstruksi selalu melibatkan aliran informasi yang konstan, dan aliran tersebut kini telah terdigitalisasi. Buku ini, kursus seperti seri Procore.org saya tentang "data dalam konstruksi," dan lainnya, semuanya menawarkan pengenalan yang relatif mudah terhadap keterampilan digital yang akan dibutuhkan.

Yang kami maksud ketika kami mengatakan pola pikir konstruksi digital, dan poin terakhir yang ingin saya sampaikan dalam buku ini, adalah bahwa teknologi hanyalah sebuah alat. Teknologi melengkapi pengalaman, penilaian, dan keterampilan yang hanya dapat diberikan oleh manusia. Anda, profesional konstruksi, adalah teknologi yang tak tergantikan. Pelajari sisanya dan manfaatkan untuk Anda.

DAFTAR PUSTAKA

Adams, A. M., & Moen, B. S. (2015). Software engineering: A practical perspective. Pearson Education.

Al-Hajj, H., & Zohdy, M. A. (2018). Software engineering and development techniques: A comprehensive guide. Wiley-IEEE Press.

Ambler, S. W. (2009). The agile database techniques: Effective strategies for the agile software developer. Wiley.

Ambler, S. W. (2012). Agile modeling: Effective practices for extreme programming and the unified process. Wiley.

Andersen, L. L., & Smith, J. T. (2016). Test-driven development: A practical guide. Springer.

Bass, L., Clements, P., & Kazman, R. (2003). Software architecture in practice (2nd ed.). Addison-Wesley.

Beck, K. (2000). Extreme programming explained: Embrace change. Addison-Wesley.

Benington, H. D. (1997). Software engineering at NASA: A case study. IEEE Software, 14(5), 92-100. https://doi.org/10.1109/52.624505

Bischof, L. J., & Sonderegger, W. (2012). Engineering software for modern systems: An integrated approach. Wiley.

Boehm, B. W. (1988). A spiral model of software development and enhancement. ACM SIGSOFT Software Engineering Notes, 11(4), 14-24. https://doi.org/10.1145/55088.55089

Boehm, B. W. (2000). Software engineering economics. Prentice-Hall.

Botta, C. A. (2007). Modeling software architectures: A case-driven approach. Wiley.

Brooks, F. P. (1975). The mythical man-month: Essays on software engineering. Addison-Wesley.

Brown, S., & Senter, C. (2009). Agile software development: Best practices for successful software projects. Springer.

Cao, Y., & Ramesh, B. (2008). Agile software development: A survey of early adopters. Information & Software Technology, 50(9-10), 572-583. https://doi.org/10.1016/j.infsof.2007.07.004

Charette, R. N. (2005). Software engineering risk management. Addison-Wesley.

Cockburn, A. (2001). Agile software development. Addison-Wesley.

Cohn, M. (2004). User stories applied: For agile software development. Addison-Wesley.

Conway, M. (2003). Software engineering practices: From concept to development. Prentice Hall.

Davis, A. M. (2005). Software engineering: A contemporary perspective. McGraw-Hill.

DeMillo, R. A., Lipton, R. J., & Perlis, A. J. (1978). Hints on software engineering. ACM Computing Surveys, 10(3), 331-348.

Dijkstra, E. W. (1976). A discipline of programming. Prentice-Hall.

Elliott, J., & McMillan, G. (2014). The software engineering body of knowledge. IEEE Computer Society.

Finkel, H. J., & Thomas, M. P. (2010). Software engineering design: Theory and practice. CRC Press.

Fowler, M. (2002). Patterns of enterprise application architecture. Addison-Wesley.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design patterns: Elements of reusable object-oriented software. Addison-Wesley.

Garlan, D., & Shaw, M. (1994). An introduction to software architecture. Advances in Software Engineering and Knowledge Engineering, 1, 1-39.

Ghezzi, C., Jazayeri, M., & Mandrioli, D. (2003). Fundamentals of software engineering. Prentice Hall.

Gill, P., & Kim, Y. (2013). Model-driven software engineering: A comprehensive review. Computer Science Review, 9(1), 22-33. https://doi.org/10.1016/j.cosrev.2013.05.001

Grady, R. B. (1997). Practical software metrics for project management and process improvement. Prentice Hall.

Hall, T. (2006). Software engineering for students: A programming approach (4th ed.). Prentice Hall.

Harel, D. (2004). Statemate: A working environment for the development of complex reactive systems. IEEE Transactions on Software Engineering, 16(4), 428-441. https://doi.org/10.1109/TSE.1990.232331

Humphrey, W. S. (1989). Managing the software process. Addison-Wesley.

Jackson, D. (2000). Software testing and analysis: Process, principles, and techniques. Wiley.

Jackson, M. A. (2001). Software requirements and specification: A lexicon of practice, principles, and prejudices. ACM Press.

Jones, C. (2000). Software engineering best practices. McGraw-Hill.

Jorgensen, P. C. (2008). Software testing: A craftsmans approach (3rd ed.). CRC Press.

Kahng, A. B., & Robbins, T. (2003). Software testing and verification. Springer.

Keil, M., & Cule, P. (2015). Managing software projects: Tools and techniques. Pearson Education.

Kroll, P., & Kruchten, P. (2003). The rational unified process: An introduction. Addison-Wesley.

Kruchten, P. (2004). The rational unified process: An introduction (3rd ed.). Addison-Wesley.

Larman, C. (2004). UML and patterns: Designing software architectures. Prentice Hall.

Lehman, M. M., & Belady, L. A. (1985). Program evolution: Processes of software change. Prentice-Hall.

Liskov, B. (2009). Program development in Java: A process-oriented approach. ACM Press.

Lister, T. (2004). Extreme programming explained: Embrace change. Addison-Wesley.

McCall, J. A., & Harlan, G. (1979). Software quality: A framework for comparison. IEEE Transactions on Software Engineering, 5(3), 129-134. https://doi.org/10.1109/TSE.1979.236715

McConnell, S. (2004). Code complete (2nd ed.). Microsoft Press.

Menasce, D. A., & Almeida, V. A. F. (2002). Capacity planning and performance modeling: From mainframes to client-server systems. Prentice Hall.

Mills, H. D., & Mockus, A. (2002). Software configuration management: A rigorous approach. Springer.

Morris, L. (2013). Practical software estimation: Functional points and use cases. Wiley-IEEE Press.

Nielson, J. (1993). Usability engineering. Academic Press.

Ockerbloom, J., & Lanza, M. (2011). Modern software engineering techniques: A case study approach. Wiley.

Oestereich, B. (2005). Software engineering with the Unified Process: A practitioner's approach. Springer.

Pressman, R. S. (2000). Software engineering: A practitioner's approach (6th ed.). McGraw-Hill.

Pressman, R. S. (2014). Software engineering: A practitioner's approach (8th ed.). McGraw-Hill.

Rajlich, V. (2005). Software evolution. Springer.

Royce, W. W. (1970). Managing the development of large software systems. Proceedings of IEEE Wescon, 1-9. https://doi.org/10.1109/IEEE.1970.1010153

Rumbaugh, J., Jacobson, I., & Booch, G. (1999). The unified modeling language reference manual. Addison-Wesley.

Sommerville, I. (2004). Software engineering (7th ed.). Addison-Wesley.

Sommerville, I. (2011). Software engineering (9th ed.). Addison-Wesley.

Sommerville, I. (2011). Software engineering (9th ed.). Addison-Wesley.

Spolsky, J. (2001). Joel on software: And on diverse and occasionally related matters. Apress.

Stepanek, P. (2009). The essential guide to software project management. Addison-Wesley.

Sterrett, D. (2010). Agile software development: Practices, principles, and techniques. McGraw-Hill.

Su, M. (2010). Software engineering and design: Principles and practice. Pearson Education.

Tharp, A. (2014). Agile software development for real-world applications. Wiley.

van der Meer, P. (2011). Software design: From analysis to design patterns. Springer.

Wang, H., & Chen, M. (2007). A framework for risk management in software development. International Journal of Software Engineering and Knowledge Engineering, 17(4), 477-495.

Wason, S., & Munro, L. (2012). Developing effective software systems: A process-driven approach. Prentice Hall.

West, D. (2006). Agile software development: A managerial perspective. Springer.

Whittaker, J. A. (2004). What is software testing? And why is it so hard? ACM Computing Surveys, 36(2), 104-113. https://doi.org/10.1145/1002966.1002967

Wirth, N. (1995). Algorithms + data structures = programs. Prentice Hall.

Zeldman, J. (2004). Designing with web standards. New Riders.

Zeldman, J. (2011). Designing with web standards. New Riders Publishing.

Zhao, X., & Shih, E. (2004). Software engineering for real-time systems. Wiley.

Dasar Konstruksi RPL

(Rekayasa Perangkat Lunak)

Dr. Budi Raharjo, S.Kom, M.Kom, MM.

BIODATA PENULIS

Dr. Budi Raharjo, S.Kom, M.Kom, MM lahir di Semarang, tanggal 22 Februari 1985. Beliau adalah Alumni dari Universitas Bina Nusantara (BINUS University) Jakarta dan juga alumni Universitas Kristen Satya wacana (UKSW) Salatiga. Dr. Budi Raharjo telah menjadi Dosen pada Universitas STEKOM pada mata kuliah Kepemimpinan (Leadership), mata kuliah Pengantar Akuntansi, Manajemen Proses, Manajemen Akuntansi dan Manajemen Resiko Bisnis. Selain sebagai dosen Universitas STEKOM, Dr. Budi Raharjo, M.Kom, MM juga mempunyai bisnis sendiri dalam bidang perhotelan dan juga sebagai wirausaha dalam bidang pemasok unggas (ayam) beku, ke berbagai kota besar, khususnya Jakarta dan sekitarnya.

Pengalaman beliau berwirausaha menjadi bekal utama dalam penulisan buku ajar yang diterbitkan oleh Yayasan Prima Agus Teknik (YPAT) Semarang. Oleh sebab itu bukunya berisi langkah langkah praktis yang mudah diikuti oleh para mahasiswa, saat mahasiswa mengikuti proses perkuliahan pada Universitas Sains dan Teknologi Komputer (Universitas STEKOM). Jabatan struktural yang di embannya saat ini adalah Wakil Rektor 1 (Akademik) Universitas STEKOM Semarang.

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 978-623-8642-50-2 (PDF)

9 786238 642502