Kelompok : 127 - Pagi

Keamanan Pengembangan Sistem dan Aplikasi

Mata Kuliah:
Proteksi dan Teknik Keamanan Sistem Informasi
(IKI-83408T)

Dosen:
Rahmat M. Samik-Ibrahim
Johny Moningka
Arrianto Mukti Wibowo

Disusun Oleh:
Tikno Riyanto - 7204000373

Daftar Isi

Sekuritas Pengembangan Sistem dan Aplikasi

Pendahuluan

Tujuan tulisan ini adalah mempelajari berbagai aspek keamanan dan kendali yang terkait pada pengembangan sistem informasi baik yang dikembangkan sendiri maupun yang dikembangkan oleh pihak ketiga (outsource). Tulisan ini akan membahas isu-isu keamanan dari sudut pandang pengembang (developer), pengguna (user), dan spesialis keamanan sistem informasi.

Dalam mengembangkan suatu sistem informasi, ada berbagai metodologi yang dapat digunakan. Setiap metodologi ini akan menguraikan tahapan-tahapan dalam pengembangan suatu sistem informasi. Adapun tujuan dari semua metodologi tersebut adalah keberhasilan mengembangkan sistem informasi dengan tepat waktu, tepat biaya dan sesuai kebutuhan yang diharapkan pengguna.

Untuk mengkaji isu-isu keamanan terkait dengan pengembangan sistem informasi maka penting untuk membahas setiap tahapan dalam pengembangan sistem informasi serta menganalisa celah-celah yang memiliki potensi untuk ditembus dari setiap tahapan tersebut. Sehingga kegagalan pengembangan sistem dapat dihindari dan informasi yang diproduksi dari sistem tersebut dapat dijamin hanya akan diakses oleh orang yang berhak.

Ruang Lingkup dan Tata urut

Ruang lingkup tulisan ini dibatasi pada proses pengembangan dan pengoperasian sistem dan isu-isu keamanan yang terkait. Adapun tulisan ini akan disusun dengan tata urut sebagai berikut :

BAB I. Proses Pengembangan Perangkat Lunak
BAB II. Model-Model Proses
BAB III. Software Capability Maturity Model (CMM)
BAB IV. Object-Oriented Systems
BAB V. Artificial intelligence systems
BAB VI. Database systems
BAB VII. Usaha Kecil Menengah (UKM) dan Pengembangan Sistem

BAB I. Tahapan Pengembangan Perangkat Lunak

Rekayasa perangkat lunak (Software engineering) dapat didefinisikan sebagai ilmu dan seni dari penspesifikasian, perancangan, pengimplementasian dan pengembangan program, pendokumentasian, dan prosedur pengoperasian sehingga komputer berguna bagi manusia. Definisi ini mengkombinasikan definisi yang populer dari definisi rekayasa (teknik) dan definisi perangkat lunak. Teknik (rekayasa)

didefinisikan sebagai penerapan ilmu dan matematik untuk merancang dan membangun artifact yang berguna bagi manusia. Perangkat lunak didefinisikan sebagai kumpulan program, dokumentasi, dan prosedur pengoperasian sehingga komputer berguna bagi manusia.

Dalam rekayasa perangkat lunak ada beberapa istilah yang perlu disamakan persepsinya. *verifikasi* didefinisikan sebagai proses membangun kepercayaan dari kaitan antara produk perangkat lunak dengan spesifikasi yang diberikan. *Validasi* adalah membangun kesesuian antara produk perangkat lunak yang dihasilkan dengan misi organisasi yang dijalankan. *Requirement* didefinisikan sebagai validasi menyeluruh spesifikasi yang dibutuhkan dari fungsi, interface, dan kinerja perangkat lunak yang dibangun. *Rancangan produk (product design)* adalah verifikasi spesikasi arsitektur perangkat keras dan perangkat lunak, struktur kontrol, dan struktur data dati produk yang dibangun.

Perangkat lunak yang berkualitas sulit diperoleh tanpa suatu proses pengembangan. Dua sasaran pokok dari pengembangan perangkat lunak adalah menghasilkan produk yang berkualitas yang memenuhi kebutuhan pengguna dan diselesaikan tepat waktu dan tepat biaya. Untuk mencegah terjadinya kegagalan mencapai sasaran tersebut, maka setiap tahapan dalam proses pengembangan perangkat lunak perlu dikaji.

Pengembangan perangkat lunak, secara umum akan melibatkan tahapan-tahapan sebagai berikut :

a. Definisi cakupan (Scope definition)

Definisi cakupan adalah tahap pertama dalam pengembangan sistem. Pada tahap ini dikaji apakah sistem yang akan dibangun cukup bernilai atau bermanfaat. Jika sistem memang penting untuk dibangun, selanjutnya ditentukan ukuran dan batasan proyek, visi proyek, limitasi atau batasan yang dihadapi, partisipan proyek yang diperlukan, dan juga anggaran dan waktu yang tersedia. Pada tahap ini melibatkan partisipan yang meliputi Pemilik sistem (sistem owner), Manajer proyek (Project manager), dan Analis sistem (System analys).

Ada beberapa hal yang dapat memicu keinginan untuk membangun sistem informasi, yakni adanya permasalahan (problem), mencari peluang (opportunity), dan adanya perintah (directive). Pada tahap ini partisipan akan membangun kesepahaman yang dituangkan dalam dokumen problem statement dan scope statement. Dokumen ini akan menjelaskan permasalahan dan cakupan permasalahan yang akan diatasi dengan adanya sistem informasi serta identifikasi kendala yang dihadapi seperti anggaran yang tersedia, waktu yang dibutuhkan, SDM yang tersedia atau yang tidak tersedia, aturan-aturan bisnis, dan standar teknologi. Dengan dokumen ini maka akan dapat mengendalikan *scope creep* baik terhadap anggaran maupun waktu pengembangan.

Berdasarkan dokumen ini maka analis sistem dapat mengatur team proyek, memperkirakan anggaran pengembangan sistem, dan mempersiapkan jadwal untuk tahapa-tahap berikutnya. Puncak dari tahap ini adalah keputusan dari pemilik sistem untuk melanjutkan proyek pengembangan sistem dimana menyetujui terhadap cakupan, anggaran dan jadwal proyek, atau mengurangi cakupan sehingga dapat mengurangi anggaran dan waktu pengembangan, atau membatalkan proyek pengembangan sistem.

Jika pemilik sistem menyetujui proyek pengembangan sistem, maka dokumentasi terpenting dan terakhir pada tahap ini adalah dokumentasi *Statement of work*. Dokumen ini mengkonsolidasikan problem statement, scope statement, dan anggaran dan jadwal untuk semua pihak yang terlibat proyek.

b. Analisa permasalahan (Problem analysis)

Pada tahap ini dilakukan analisa terhadap sistem yang ada, tanpa memperhatikan apakah sistem tersebut menggunakan teknologi informasi atau tidak. Dengan mempelajari sistem yang ada dan menganalisa temuan-temuan yang diperoleh maka akan memberikan pemahaman yang lebih baik kepada team mengenai permasalahan yang memicu adanya proyek ini. Hal penting yang harus diperhatikan analis sistem bahwa keuntungan yang diperoleh dari pemecahan sistem ini harus melebihi biaya yang dibutuhkan untuk membangun sistem. Partisipan yang terlibat pada tahap ini meliputi Pemilik sistem, Pengguna sistem Sistem user), Manajer proyek, dan Analis sistem.

Dokumen yang dihasilkan pada tahap ini adalah *system improvement objectives* yang diperoleh setelah memahami proses bisnis. Dokumen ini menjelaskan kriteria bisnis dimana sistem baru yang dibangun ini akan dievaluasi, bukan menjelaskan input, output, atau proses dari sistem tersebut. Misalnya, besarnya pengurangan waktu proses antar bagian. Sasaran-sasaran peningkatan yang akan dihasilkan oleh sistem disampaikan kepada Pemilik maupun pengguna sistem tertulis maupun lisan. Dokumentasi sistem yang ada (sering disebut ‘as-is business model’) biasa disertai gambaran ketidakefisienan, bottleneck, atau permasalahan lain yang terkait dengan proses bisnis.

Pemilik sistem setelah mereview hasil-hasil temuan bisa jadi menerima atau menolak sasaran peningkatan sistem yang direkomendasikan. Sehingga, akhir dari tahap ini adalah salah satu dari hal-hal berikut :

- proyek dibatalkan jika permasalahan dianggap kurang bernilai untuk dipecahkan.
- Proyek disetujui untuk dilajutkan pada tahap berikutnya.
- Cakupan proyek dikurangi atau ditambah (dengan perubahan anggaran dan jadwal) dan proyek dilanjutkan ke tahap berikutnya.

c. Analisa kebutuhan (Requirement analysis)

Tahap analisa kebutuhan mendefinisikan dan melakukan prioritasisasi kebutuhan bisnis. Analis sistem mendekati pengguna untuk mengetahui apa yang dibutuhkannya, atau apa yang diinginkannya terhadap sistem yang akan dikembangkan. Tahap ini belum membahas teknologi yang akan digunakan atau teknik implementasinya.

Analisa kebutuhan merupakan tahap penting dalam pengembangan sistem. Kesalahan atau kelalaian dalam analisa kebutuhan ini berakibat pada ketidakpuasan pengguna terhadap sistem yang dihasilkan, sehingga memerlukan modifikasi yang membutuhkan waktu dan biaya tersendiri. Adapun partisipan yang terlibat dalam tahap ini meliputi Pengguna sistem, Analis sistem, dan Manajer proyek. Sedangkan Perancang sistem belum dilibatkan untuk menghindari atensi yang terlalu dini untuk solusi teknologi.

Dokumen yang dihasilkan pada tahap ini adalah *Business Requirement Statement*. Dokumen ini mengupas kebutuhan pengguna dan tingkat prioritasnya. Untuk dapat mengidentifikasi dengan baik dan benar mengenai kebutuhan pengguna beserta prioritasnya ini, maka analis sistem harus bekerja secara erat dengan pengguna sistem. Kebutuhan penting dilakukan prioritasisasi. Salah satu kegunaan adalah untuk melakukan *rescope* yakni penentuan cakupan yang baru terhadap proyek karena keterbatasan waktu yang tersedia. Adapun Informasi kebutuhan beserta prioritasnya ini dikumpulkan oleh analis sistem dengan berbagai cara, seperti interview, kuisioner, maupun pertemuan.

Tahapan ini penting mendapat perhatian karena salah satu keluhan paling umum terhadap sistem baru adalah sistem tidak benar-benar memenuhi kebutuhan pengguna. Hal ini terjadi karena perancang dan pembangun sistem terjebak ke solusi teknis sebelum benar-benar memahami kebutuhan pengguna. Oleh karena itu, kerjasama antara analis sistem dan pengguna sistem dalam mendefinisikan dan mendokumentasikan secara lengkap dan akurat dari kebutuhan bisnis adalah sdangat penting, sebelum bicara mengenai teknologi yang akan digunakan.

d. Rancangan logis (Logical design)

Tahapan rancangan logis adalah menterjemahkan kebutuhan bisnis kedalam system model dalam bentuk gambar. Contoh umum dari system model adalah Data Flow Diagram (DFD). Istilah rancangan logis diinterpretasikan sebagai *technology independent*, yakni tidak terikat teknologi tertentu. Ini berarti bahwa system model menggambarkan sistem yang tidak terikat oleh solusi teknologi tertentu. Ia bisa diimplementasikan dengan teknologi sesuai pertimbangan kita.

Metodologi berbeda memerlukan dan merekomendasikan jumlah dan tingkat yang berbeda terhadap pemodelan sistem atau rancangan sistem. Metodologi prescriptive seperti structured analysts and design, information

engineering, dan Rational Unified Process biasanya memerlukan dimana banyak tipe model sistem digambarkan pada beragam tingkat rincian. Sementara itu, metodologi agile seperti architected rapid application development dan extreme programming merekomendasikan “just enough modeling”.

Partisipan yang terlibat dalam tahap ini meliputi analis sistem yang menggambarkan model, pengguna sistem yang memvalidasi model, manajer proyek yang menjamin pemodelan memenuhi standar. Adapaun gambar yang dihasilkan meliputi *Logical Data Model* yang menggambarkan data dan informasi yang dibutuhkan, *Logical Process Model* yang menggambarkan proses bisnis yang dibutuhkan, *LogicalInterface Model* yang menggambarkan antar muka sistem dan bsinis yang dibutuhkan.

Pada tahap ini menghasilkan *Logical system model adn Specification*. Sesuai dengan metodologi yang digunakan, maka tingkat kerincian dari spesifikasinya akan bermacam-macam.

e. Analisa keputusan (Decision analysis)

Berdasarkan kebutuhan sistem dan model sistem, ada sejumlah cara cara untuk merancang sistem informasi baru yang memenuhi kebutuhan tersebut. Beberapa pertanyaan yang terkait dengan hal ini meliputi :

- Seberapa banyak sistem akan diotomatisasi dengan sistem informasi ?
- Apakah perangkat lunak akan dibeli dari pihak ketiga atau dibangun sendiri ?
- Apakah sistem yang dibangun untuk internal network ataukah berbasis Web ?
- Apakah teknologi informasi (yang kemungkinan muncul) berguna untuk aplikasi ini ?

Tahapan analisa keputusan merupakan tahapan untuk mengkaji jawaban dari pertanyaan-pertanyaan tersebut. Sasaran dari tahap ini adalah mengidentifikasi solusi-solusi kandidat teknis, menganalisa kelayakan dari masing-masing solusi kandidat tersebut, dan merekomendasikan kandidat solusi terbaik untuk ditindaklanjuti ketahap perancangan fisik.

Analisa keputusan merupakan tahapan transisi dari analisa yang konsentrasi pada pengguna sistem dengan masalah bisnisnya ke rancangan sistem yang konsentrasi ke perancang sistem dan pembangun sistem. Perancang sebagai ahli teknis untuk teknologi tertentu, mulai memainkan peranannya dalam tahap ini bersama-sama dengan pengguna sistem dan analis sistem. Keputusan yang dibuat pada tahap ini memperhatikan teknologi yang akan digunakan sebagai bagian dari arsitektur aplikasi.

Team proyek mengumpulkan ide dan opini untuk rancangan teknis dan implementasi dari bermacam-macam audien termasuk vendor-vendor perangkat lunak. Solusi-solusi kandidat diidentifikasi dan dikelompokkan menurut berbagai macam kriteria. Perlu diperhatikan bahwa beberapa organisasi modern memiliki teknologi informasi dan standard arsitektur yang membatasi sejumlah kandidat solusi yang dipertimbangkan dan dianalisa.

Setelah solusi-solusi kandidat diidentifikasi, selanjutnya dievaluasi kelayakannya yang mencakup kelayakan teknis, operasional, ekonomi, jadwal, dan resiko. Kelayakan teknis mengevaluasi apakah solusi secara teknis dapat diterapkan, dan apakah staff memiliki keahlian teknis untuk merancang dan membangun solusi tersebut. Kelayakan operasional mengevaluasi apakah solusi memenuhi kebutuhan penguna, bagaimana solusi merubah lingkungan kerja pengguna, dan bagaimana perasaan pengguna dengan solusi tersebut. Kelayakan ekonomi mengevaluasi apakah keuntungan yang diperoleh dari solusi yang ditawarkan memberi manfaat yang lebih besar dari biaya untuk pengembangannya. Kelayakan jadwal mengevaluasi apakah solusi dapat dirancang dan diimplementasikan dalam periode waktu yang disediakan. Kelayak resiko mengevaluasi seberapa besar tingkat keberhasilan implementasi denganmenggunakan pendekatan dan teknologi yang dipilih.

Team proyek akan melihat solusi yang paling layak, yakni solusi yang memberikan kombinasi terbaik dari kelayakan secara teknis, operasional, ekonomis, jadwal, dan resiko. Suatu solusi kandidat mungkin akan memberikan kelayakan terbaik pada kriteria tertentu. Namun, solusi kandidat yang dipilih adalah yang memberikan kelayakan terbaik untuk semua kriteria.

Semua kegiatan analisa diatas, dituangkan dalam dokumentasi yang disebut S*ystem Proposal*, dokumentasi yang harus dihasilkan pada tahap ini. Dokumentasi ini biasanya dilengkapi juga dengan *Application Architecture*. Ada beberapa kemungkinan tanggapan dari Pemilik sistem terhadap system proposal ini, yaitu :

- Menyetujui dan mendanai proposal sistem untuk dilanjutkan ke tahap perancangan dan konstruksi (termasuk kemungkinan memodifikasi anggaran dan penjadwalan jika merubah cakupan secara signifikan).
- Menyetujui dan mendanai satu dari alternatif solusi kandidat.
- Menolak semua solusi kandidat dan membatalkan proyek atau kembali dengan rekomendasi baru.

Bentuk lain dari proposal system adalah *Request For Systsem Proposal (RFP)*. RFP dibutuhkan sebagai rekomendasi pembelian perangkat keras ataupun perangkat lunak sebagai solusi yang dipilih. Jadi ini berlawanan dengan sistem yang dibangun sendiri (in-house).

f. Perancangan fisik (Phisical design) dan integrasi

Kegiatan perancangan sistem baru mengacu pada system proposal yang dihasilkan pada tahap analisa keputusan. Kegunaan dari rancangan fisik sistem ini adalah untuk mentransformasikan kebutuhan bisnis (yang digambarkan dalam logical system model) ke dalam *Phisical Design Spesification* yang selanjutnya akan diguakan sebagai panduan dalam membangun sistem. Pada tahap ini akan diurai secara lebih detil mengenai teknologi yang akan digunakan pada sistem baru nanti. Perancangan fisik ini dibatasi oleh *architectural model* yang sudah disetujui pada tahapan sebelumnya. Perancangan juga memerlukan ketaatan pada standard perancangan teknis internal untuk menjamin kelengkapan, usabilitas, reliabilitas, performa, dan kualitas.

Perancangan fisik bertolak belakang dengan perancangan logis. Perancangan logis berurusan secara eksklusif dengan kebutuhan bisnis yang 'bebas' dari solusi teknis, sedangkan perancangan fisik menggambarkan solusi teknis yang spesifik. Tahap ini berkonsentrasi pada melihat sistem dari sudut pandang teknologi, yang meliputi spesifikasi rancangan database fisik (Physical database design specification), spesifikasi rancangan perangkat lunak dan proses bisnis fisik (Physical business process and software design specification), dan spesifikasi antarmuka sistem dengan pengguna fisik (Physical user and system interface specification). Partisipan kunci yang terlibat dalam tahap ini adalah perancang sistem dan analis sistem. Namun, dalam aspek tertentu dari perancangan, seperti rancangan tampilan (screen) dan work flow, harus melibatkan pengguna sistem.

Dalam hal perancangan fisik ini ada dua filosofi yang sangat berbeda, yakni perancangan dengan spesifikasi (*design by specification*) dan perancangan dengan prototipe (*design by prototyping*). Perancangan dengan spesifikasi mendeskripsikan spesifikasi terinci dan model sistem fisik sebagai suatu rangkaian cetak biru yang tertulis untuk dikonstruksi. Sedangkan, perancangan dengan prototipe menyelesaikan perancangan dengan membuat prototipe dari sistem yang belum lengkap, selanjutnya dibangun dan diperbaiki berdasarkan umpan balik dari pengguna ataupun perancang lain. Dalam prakteknya, kombinasi perancangan dari kedua konsep ini seringkali dilakukan dalam pengembangan suatu sistem.

Organisasi dalam mengembangakan suatu sistem informasi baru umumnya tidak terisolasi dari sistem lain yang sudah ada. Konsekuensinya, dalam merancang sistem baru juga harus memperhatikan dan merefleksikan integrasi sistem. Sistem baru harus terintegrasi baik dengan sistem informasi lain maupun dengan proses-proses bisnis itu sendiri. Integrasi biasanya direfleksiakan dalam spesifikasi rancangan dan model sistem fisik.

Dokumentasi yang dihasilkan pada tahap ini meliputi *Physical design model and specification, Design prototype,* dan *Redesigned business process*.

Pemilik sistem jarang sekali melakukan pembatalan proyek setelah tahap perancangan fisik. Hal mungkin dilakukan adalah memodifikasi cakupan untuk disesuaikan dengan anggaran dan jadual, atau memodifikasi jadual dan anggaran untuk disesuaikan dengan cakupan yang diinginkan.

g. Konstruksi dan pengujian (Construction and testing)

Tahap konstruksi dan pengujian dilakukan setelah diperoleh spesifikasi dan model rancangan fisik, dan atau rancangan prototipe. Dokumen pokok yang dihasilkan pada tahap ini adalah *Functional system* yang siap untuk diimplementasikan. Kegunaaan tahap konstruksi dan pengujian ada dua. Pertama, untuk membangun dan menguji bahwa sistem yang dibangun memenuhi kebutuhan bisnis dan sesuai spesifikasi rancangan fisik. Kedua, untuk mengimplementasikan antarmuka antara sistem baru dengan sistem yang ada. Dalam tahap ini juga bisa mencakup instalasi dari software yang dibeli.

Hal-hal yang harus dikonstruksi atau diinstall oleh team proyek ada tiga. Satu, *Database* yang meliputi *Online Transaction Processing (OLTP)* database untuk mendukung transaksi bisnis harian, *Operational Data Store (ODS)* untuk mendukung reporting dan query harian, dan *data warehouse* untuk mendukung kebutuhan data dalam analisis dan pengambilan keputusan. Dua, *Commercial Software Package* dan/atau *Custom-Built Software*. Paket diinstal dan dikustomisasi sesuai kepentingan. Program aplikasi dibangun sesuai rancangan fisik dan/atau prototype yang dihasilkan pada tahap sebelumnya. Baik perangkat lunak paket maupun kustom harus diuji secara seksama. Tiga, *User and system interface*. Antarmuka user (seperti windows dan antarmuka web) harus diuji untuk menjamin usabilitas dan stabilitas. Antarmuka sistem ke sistem harus dibangun atau diimplementasikan menggunakan teknologi integrasi aplikasi. Middleware, suatu jenis dari perangkat lunak sistem, selalu digunakan untuk mengintegrasikan teknologi antarmuka, perangkat lunak, dan basis data yang berbeda-beda.

Partisipan yang terlibat dalam tahap ini meliputi pembangun sistem (system builder), analis sistem, dan manajer proyek. Keterlibatan perancang sistem (system designer) dalam tahap ini lebih untuk klarifikasi spesifikasi rancangan sistem.

Satu aspek terpenting dari tahap konstruksi adalah pelaksanaan pengujian. Pengujian dilakukan baik per komponen dari sistem maupun sistem secara menyeluruh. Setelah diuji maka siap dilakukan tahap berikutnya yakni instalasi dan penyerahan (*installation and delivery*).

h. Instalasi dan penyerahan (Installation and delivery)

Setelah sistem selesai dibangun dan sudah melewati pengujian maka pengguna sistem akan meninggalkan cara bisnis yang selama ini diterapkan. Analis sistem harus menyiapkan proses transisi yang halus dari sistem lama ke sistem baru dan membantu dan mendampingi pengguna dalam memulai pengunaan sistem baru ini.

Functional system yang dihasilkan pada tahap konstruksi dan pengujian adalah masukan kunci untuk tahap instalasi dan penyerahan. Yang dihasilkan pada tahap ini adalah *Operational system.* Pengembang sistem menginstal sistem dari lingkungan pengembangannya ke lingkungan operasioal. Analis sistem harus melatih pengguna, membuat petunjuk pengoperasian, mengkonversi file dan database yang digunakan saat ini ke dalam database yang baru, dan menjalankan pengujian akhir sistem.

Untuk dapat melaksanakan transisi secara halus kedalam sistem baru, rencana konversi harus dipersiapkan. Rencana konversi ini bisa dengan cara menggantikan sistem lama dengan sistem baru dalam periode waktu tertentu, atau menjalankan sistem lama dan sistem baru secara paralel hingga sistem baru dipastikan diterima untuk menggantikan sistem lama.

Tahap instalasi dan penyerahan juga mencakup pelatihan individu yang akan menggunakan sistem dan pengembangan dokumentasi untuk membantu pengguna sistem. Tahap implementasi biasanya mencakup beberapa bentuk ‘post-audit review’ untuk mengukur keberhasilan proyek sistem secara lengkap.

BAB II. Model-Model Proses

Model proses merupakan peta jalan (roadmap) yang sangat berguna bagi perekayasa perangkat lunak untuk menghasilkan produk yang berkualitas tinggi. Ada banyak model proses dengan berbagai kelebihan dan kekurangan masing-masing yang akan dibahas lebih lanjut. Pengembang sistem memilih suatu model proses untuk mengembangkan sistem dan mengikutinya.

Model proses penting dalam pengembangan sistem karena akan menyediakan stabilitas, kontrol, dan organisasi sehingga proses rekayasa perangkat lunak terhindar dari chaos. Model proses mengarahkan team pengembang perangkat lunak melalui kerangka kerja yang disusun dalam suatu aliran kerja yang bisa linear, incremental, atau evolusioner. Terminologi dan detil dari tiap-tiap model proses bisa berbeda, namun aktivitas framework tetap sama.

a. Waterfall Model

Waterwall model, yang sering disebut juga classic life cycle, menyampaikan suatu pendekatan yang berurutan untuk pengembangan perangkat lunak. Pengembangan dimulai dari spesifikasi kebutuhan dan berlanjut dengan perencanaan, pemodelan, konstruksi, dan penyerahan.

Waterfall model adalah paradigma tertua dalam rekayasa perangkat lunak. Namun, pada dua dekade terakhir dengan banyaknya kritik terhadap model proses ini telah memunculkan keraguan dari para pengguna model ini mengenai kehandalan model proses ini. Diantara permasalahan yang kadang-kadang muncul ketika model ini diterapkan adalah sebagai berikut :

- Proyek-proyek di dunia nyata jarang yang mengikuti alur berurutan seperti yang dikemukakan model ini. Meskipun iterasi dapat diakomodasi dalam model linear seperti ini, namun tidak dilakukan secara langsung. Sebagai akibatnya, perubahan-perubahan yang terjadi dapat menyebabkan kebingungan bagi anggota team pengembangan.

- Seringkali pengguna/customer kesulitan menyampaikan semua kebutuhannya secara jelas. Sedangkan, model ini mensyaratkan bahwa semua kebutuhan customer harus disampaikan secara jelas pada tahap awal proyek dikerjakan

- Customer harus memiliki kesabaran, karena hasil proses pengembangan perangkat lunak tidak segera dapat dilihat oleh customer.

Suatu analisa menarik dari proyek nyata yang mengikuti model proses waterfall bahwa model ini bisa membawa kearah ‘blocking state’ atau status terblok. Status terblok ini terjadi dimana beberapa anggota team harus menunggu anggota lain team ini untuk menyelesaikan tugas yang ada kaitannya. Dimana, waktu yang diperlukan untuk menunggu ini bisa melebihi waktu produktif yang diperlukan untuk mengerjakan tugasnya. Status terblok ini cenderung terjadai pada tahap awal dan akhir dari model ini.

b. Incremental Model

Incremental model memadukan elemen-elemen waterfall model yang diterapkan dalam suatu cara yang berulang-ulang. Incremental model menerapkan urutan linear dalam suatu cara yang bergiliran. Masing-masing urutan linear menghasilkan perangkat lunak tambahan. Sebagai contoh, perangkat lunak word-processing dikembangkan dengan paradigma incremental memberikan fungsi-fungsi produksi dokumen, editing, dan pengelolaan file dasar

pada ‘increment’ pertama. Kemampuan produksi dokumen dan editing yang lebih komplek dihasilkan pada increment kedua. Kemampuan pengejaan dan pemeriksaan tata tulis dihasilkan pada increment ketiga. Kemampuan tata letak halaman lanjutan dihasilkan pada increment keempat.

Ketika incremental model digunakan, maka produk pertama adalah selalu *core product* (produk inti). Ini adalah kebutuhan dasar yang harus diselesaikan, dan fasilitas-fasilitas pendukung belum diselesaikan. Produk inti digunakan oleh customer atau menjalani evaluasi terinci. Sebagai hasil dari penggunaan dan atau evaluasi adalah pembuatan rencana untuk produk tambahan berikutnya. Perencanaan mencakup modifikasi produk inti menjadi lebih baik memenuhi kebutuhan customer dan memberikan fasilitas dan fungsi tambahan. Proses ini diulang-ulang hingga produk yang lengkap dihasilkan.

Incremental model sekilas seperti prototyping dimana proses dilakukan berulang-ulang. Namun, sesungguhnya berbeda dengan prototyping dimana incremental model fokus pada penyerahan produk operasional untuk setiap produk menyediakan kapabilitas untuk melayani pengguna dan menyediakan platform bagi pengguna untuk melakukan evaluasi.

Pengembangan secara incremental sangat berguna ketika staf yang dimiliki tidak mencukupi untuk implementasi menyeluruh dari batas waktu yang tersedia untuk menyelesaikan proyek. Produk awal dapat diimplementasikan dengan jumlah orang yang lebih sedikit. Jika produk inti diterima dan disetujui, staf tambahan (jika diperlukan) dapat ditambahkan untuk mengimplementasikan produk tambahan berikutnya.

c. RAD Model

Rapid Application Development (RAD) model adalah termasuk model proses pengembangan perangkat lunak incremental, hanya saja menekankan pada siklus pengembangan yang singkat. Model RAD adalah adaptasi ‘kecepatan tinggi’ dari waterfall model, yang mana kecepatan pengembangan dicapai melalui pendekatan konstruksi berbasis komponen. Jika kebutuhan sistem dimengerti dengan baik dan cakupan proyek dibatasi, maka proses RAD memungkinkan team pengembang menciptakan “fully functional system” dalam periode waktu yang sangat singkat.

Seperti model proses lain, pendekatan RAD memetakan kedalam aktivitas -aktivitas kerangka kerja umum sebagaimana ditunjukkan sebelumnya. Komunikasi bekerja untuk memahami permasalahan bisnis dan karakteristik informasi yang harus diakomodasi perangkat lunak. Perencanaan adalah esensiil karena ada banyak team pengembang yang bekerja bersama-sama untuk fungsi-fungsi sistem yang berbeda-beda. Pemodelan mencakup tiga tahap utama yakni pemodelan bisnis, pemodelan data, dan pemodelan proses. Pada pemodelan ini

juga membangun rancangan sebagai dasar untuk aktivitas konstruksi. Konstruksi mengutamakan penggunaan komponen software yang sudah ada sebelumnya.

Sebagimana model proses lain, RAD model juga memiliki kelemahan. Beberapa kelemahan dari RAD model adalah sebagai berikut :

- Untuk sistem yang luas, namun proyek dengan skalabilitas. RAD memerlukan sumber daya manusia yang mencukupi untuk bisa membagi menjadi sejumlah team RAD yang tepat.
- Jika pengembang dan customer tidak memiliki komitmen kuat untuk menyelesaikan sistem dalam waktu singkat maka proyek RAD akan gagal.
- Jika sistem tidak dapat dimodularkan secara tepat, pengembangan komponen-komponen penting untuk RAD akan menjadi masalah.
- Jika tingginya performa menjadi isu, dan ini dicapai melalui penyesuaian antarmuka ke komponen sistem, maka pendekatan RAD ada kemungkinan tidak bekerja.
- RAD tidak sesuai ketika resiko teknis tinggi. Misalnya, ketika suatu aplikasi baru menuntut penggunaan teknologi baru.

d. Prototyping

Customer seringkali menjelaskan sekumpulan sasaran umum perangkat lunak, namun tidak mengidentifikasi kebutuhan input, proses, atau output. Pada kasus lain, pengembang mungkin tidak yakin akan efisiensi dari suatu algoritma, adaptabilitas dari suatu sistem operasi, atau bentuk yang akan diambil dalam interaksi manusia-mesin. Dalam situasi seperti ini maupun situasi lain, paradigma prototyping bisa memberikan pendekatan terbaik.

Meskipun prototyping dapat digunakan sebagai model proses berdiri sendiri (standalone), namun lebih sering digunakan sebagai teknik yang diimplementasikan bersama dengan model-model yang lain. Tanpa memperhatikan cara bagaimana model ini dipakai, paradigma prototyping membantu pengembang dan pengguna untuk memiliki pemahaman yang lebih baik tentang apa yang akan dibangun ketika kebutuhan yang diinginkan tidak diuraikan secara jelas.

Paradigma prototyping diawali dengan komunikasi. Pengembang dan pengguna bertemu dan mendefinisikan sasaran-sasaran menyeluruh dari perangkat lunak yang akan dibangun, mengidentifikasi kebutuhan apa saja yang diinginkan. Iterasi prototyping direncanakan secara cepat, demikian juga pemodelan dalam bentuk rancangan segera dibuat. Perancangan yang cepat berfokus pada penggambaran aspek-aspek perangkat lunak yang akan dilihat oleh pengguna, seperti tampilan antarmuka pengguna dengan sistem, atau format

tampilan output. Rancangan yang cepat ini akan membawa kearah pembuatan program (konstruksi) dari prototype. Prototype diserahkan dan dievaluasi oleh pengguna. Umpan balik dari pengguna digunakan untuk memperbaiki kriteria kebutuhan perangkat lunak. Hal ini dilakukan berulang-ulang dimana prototype disesuaikan untuk memenuhi kebutuhan pengguna, sementara pada saat yang sama pengembang memiliki pemahaman yang lebih baik mengenai apa yang diinginkan pengguna untuk dipenuhi.

Secara ideal, prototype adalah suatu mekanisme untuk mengidentifikasi kebutuhan dari perangkat lunak yang akan dihasilkan. Pada saat prototype ini dikembangkan, pengembang berusaha menggunakan program atau took yang ada, seperti report generator, windows manager, yang memungkinkan prototype dibuat secara cepat. Prototype berlaku sebagai sistem pengenal, bukan sebagai sistem yang benar-benar dihasilkan untuk dioperasionalkan.

Adalah benar bahwa banyak pengguna dan pengembang yang menyukai model prototyping. Pengguna dapat merasakan sistem yang akan diwujudkan, dan pengembangan dapat membangun sesuatu dengan segera. Namun, prototyping model memiliki permasalahan untuk alasan-alasan sebagai berikut :

- Pengguna melihat bahwa apa yang muncul dan dilihat dari prototype adalah perangkat lunak yangk akan dioperasionalkan. Pengguna tidak menyadari bahwa prototype tersebut belum dibangun dengan memperhatikan kualitas perangkat lunak beserta maintainabilitasnya. Jika disampaikan bahwa produk yang akan dioperasionalkan harus dibangun ulang sehingga produk memiliki kualitas tinggi dan dapat dipelihara dengan baik, maka pengguna mengeluh dan bahkan meminta beberapa perbaikan unuk turut diterapkan dalam sistem yang akan dioperasionalkan.

- Pengembang sering melakukan kompromi dalam implementasi dengan maksud agar prototype segera terwujud dengan segera. Prototype dibangun menggunakan sistem operasi dan bahasa pemrograman yang tidak tepat hanya karena yang diketahui dan tersedia. Atau, suatu algoritma yang tidak efisien diimplementasikan agar segera dapat mendemonstrasikan kemampuan. Setelah suatu waktu, pengembang bisa menjadi nyaman dengan pilihan-pilihan tersebut dan melupakan bahwa itu semua sebenarnya tidak tepat. Pilihan-pilihan yang jauh dari ideal terlanjur menjadi bagian integral dari sistem yang dibangun tersebut.

Meskipun permasalahan ada, prototyping dapat menjadi model yang efektif untuk rekayasa perangkat lunak. Kuncinya adalah aturan permainan harus dijelaskan di awal proyek, bahwa pengembang dan pengguna harus memiliki kesepahaman bahwa prototype dibuat sebagai sarana untuk mendefinisikan

kebutuhan. Sedangkan perangkat lunak sesungguhnya dibangun dengan berdasarkan kualitas.

e. Spiral Model

Spiral model memadukan prototyping model dengan waterfall model. Sistem dikembangkan dengan menggabungkan antara langkah-langkah pengembangan sistem yang menekankan pada pengulangan-pengulangan dengan cara pengembangan sistem yang menekankan pada sistematika dan kontrol.

Boehm, pencetus spiral model, menjelaskan bahwa spiral model adalah model proses yang dipicu oleh resiko, dimana digunakan untuk memandu banyak stakeholder bersama-sama membangun perangkat lunak sistem. Model ini memiliki dua ciri utama. Satu, suatu pendekatan perputaran (cyclic) untuk peningkatan pertumbuhan tingkat definisi dan implementasi sistem, sementara penurunan tingkatannya adalah resiko. Dua, sekumpulan dari ‘anchor point milestones’ untuk menjamin komitmen stakeholder pemenuhan terhadap kelayakan dan kemanfaatan dari solusi sistem.

Dengan menggunakan spiral model, perangkat lunak dikembangkan dalam suatu serangkaian hasil secasra evolusioner. Pada iterasi awal, hasil kemungkinan berupa model tulisan atau prototype. Pada iterasi selanjutnya, hasil yang lebih lengkap dari sistem yang dibangun secara meningkat akan dihasilkan.

Spiral model dibagi kedalam sekumpulan aktivitas kerangka kerja yang didefinisikan oleh team pengembang sistem. Sebagai ilustrasi akan digunakan aktivitas kerangka kerja umum seperti yang sudah dibahas terdahulu. Masing-masing kerangka kerja menggambarkan satu segmen dari jalur spiral. Sebagaimana proses evolusioner ini berawal, team pengembang mengerjakan aktivitas yang dinyatakan secara tidak langsung oleh suatu sirkuit sekeliling spiral searah jarum jam, dimulai dari pusat. Resiko dipertimbangkan sebagai setiap revolusi yang dibuat. Anchor point milestones adalah sebagai suatu kombinasi hasil pekerjaan dan kondisi yang dicapai sepanjang jalur spiral.

Sirkuit pertama sekeliling spiral dapat berupa hasil pengembangan spesifikasi sistem. Sirkuit berikutnya mengitari spiral bisa digunakan untuk mengembangkan suatu prototype dan selanjutnya secara progresif adalah versi-versi yang lebih komplit dan kompleks dari perangkat lunak yang dikembangkan. Setiap hasil melalui wilayah perencanaan dengan akibat penyesuaian rencana proyek. Biaya dan jadwal disesuaikan berdasarkan umpan balik yang diterima dari pengguna setelah diserahkan. Dalam hal ini, manajer proyek melakukan penyesuaian perencanaan sebanyak iterasi yang diperlukan untuk menyelesaikan proyek.

Tidak seperti model proses lain yang berakhir ketika perangkat lunak diserahkan, spiral model tidak berakhir sepanjang hidup dari perangkat lunak komputer tersebut. Oleh karena itu, Sirkuit pertama mengitari spiral bisa menggambarkan suatu “konsep proyek pengembangan” yang bermula dari inti spiral dan berlanjut dengan iterasi berulang-ulang hingga konsep pengembangan selesai. Jika konsep dikembangkan kedalam produk nyata, proses berlanjut ke sebelah luar spiral dan mulai suatu “proyek pengembangan produk baru”. Produk baru akan berkembang melalui sejumlah iterasi mengelilingi spiral. Selanjutnya, sirkuit mengitari spiral mungkin digunakan untuk menggambarkan suatu “proyek peningkatan produk”. Intinya, spiral akan tetap beroperasi hingga sistem itu sendiri dihentikan. Ada suatu saat ketika proses terbengkelai, namun kapan saja suatu perubahan dimulai, maka proses memulainya tidak dari awal lagi, tetapi dari titik masuk yang sesuai, misalnya peningkatan produk.

Spiral model merupakan suatu pendekatan realistis untuk pengembangan sistem skala besar. Hal ini karena sistem berkembang sebagaimana proses berkembang, pengembang dan pengguna memiliki pemahaman yang lebih baik dan reaksi terhadap resiko pada level evolusioner. Spiral model menggunakan prototyping sebagai sarana mengurangi resiko, namun yang lebih penting, pengembang dimungkinkan menggunakan pendekatan prototyping pada banyak tahap dalam evolusi produk. Model ini juga memelihara pendekatan sistematis seperti yang dianjurkan model waterfall namun memadukannya kedalam kerangka kerja iteratif yang lebih realistis merefleksikan dunia nyata. Spiral model meminta langsung pertimbangan resiko teknis dari semua tahap proyek, dan jika diterapkan secara tepat maka akan mengurangi resiko sebelum benar-benar menjadi permasalahan.

BAB III. Software Capability Maturity Model (CMM)

Organisasi pengembang sistem yang semakin matang maka akan semakin tepat waktu dalam menyelesaikan proyek dan biaya yang semakin rendah, sementara itu kualitas dan produktivitas semakin meningkat. Software Engineering Institute (SEI) di Carnegie Mellon University telah mengamati dan mengukur fenomena ini dan mengembangkan Capability Maturity Model (CMM) untuk membantu seluruh organisasi pengembang sistem dalam memperoleh manfaat/keuntungan ini. Model ini sudah dikembangkan dan diikuti secara luas, baik dikalangan industri maupun pemerintahan.

Kerangka kerja CMM diharapkan untuk membantu organisasi meningkatkan kematangan proses pengembangan sistem mereka. CMM disusun dalam lima tingkat kematangan, yaitu :

- Level 1 (initial). Ini sering disebut anarki atau kacau (chaos). Pada level ini, proyek pengembangan sistem mengikuti proses yang tidak konsisten. Maing-masing team pengembang menggunakan metode dan tool sendiri. Keberhasilan dan

kegagalan pengembangan sistem sebanding dengan ketrampilan dan pengalaman dari team. Proses tidak dapat diprediksi dan tidak dapat diulang. Proyek biasanya menemukan banyak krisis dan sering melebihi anggaran yang direncanakan dan penyelesaian proyek tidak tepat waktu. Dokumentasi dibuat secara sporadis dan tidak konsisten dari satu proyek ke proyek berikutnya. Hal ini memunculkan permasalahan dalam pemeliharaan sistem. Hampir semua organisasi memulai dari level 1.

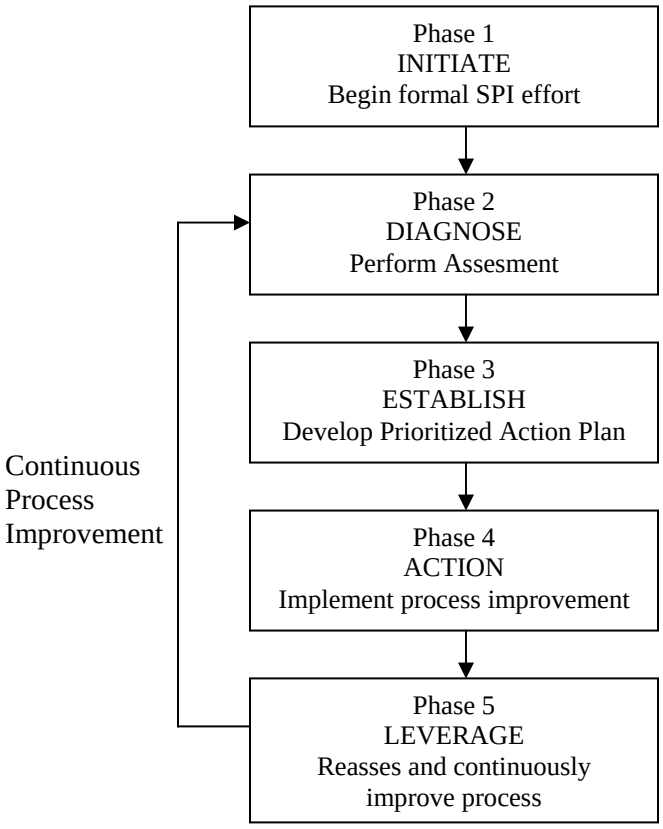
- Level 2 (Repeatable). Pada level ini, proses dan praktek pengelolaan proyek dibuat untuk mengikuti biaya proyek, jadwal, dan fungsionalitas. Fokus proyek adalah pada manajemen proyek Suatu proses pengembangan sistem selalu diikuti, namun bisa berbeda dari satu proyek ke proyek berikutnya. Keberhasilan dan kegagalan pengembangan sistem sebanding dengan ketrampilan dan pengalaman dari team, namun usaha-usaha yang dilakukan dibuat untuk mengulangi keberhasilan proyek sebelumnya. Praktek pengelolaan proyek yang efektif menjadi pondasi untuk standarisasi proses pada level berikutnya.

- Level 3 (Defined). Pada level ini, suatu proses pengembangan sistem standard (sering disebut metodologi) diterapkan. Seluruh proyek menggunakan versi yang ditetapkan untuk mengembangkan dan memelihara perangkat lunak dan sistem informasi. Sebagai hasil dari penggunaan proses yang sudah distandarisasi untuk semua proyek, maka setiap proyek memberikan hasil dan dokumentasi yang konsisten dan kualitas yang tinggi. Oleh karena itu, proses pada level ini adalah stabil, dapat diprediksi, dan dapat diulang.

- Level 4 (Managed). Pada level ini, sasaran-sasaran yang terukur untuk kualitas dan produktivitas dibangun. Ukuran-ukuran terinci dari proses pengembangan sistem standard dan kualitas produk dikumpulkan secara teratur dan disimpan dalam suatu database. Ini adalah suatu usaha untuk meningkatkan manajemen proyek individual berdasarkan data yang dikumpulkan ini. Sehingga, manajemen menjadi lebih proaktif daripada reaktif untuk permasalahan pengembangan sistem (seperti biaya yang berlebihan, cakupan yang berubah-ubah, jadwal yang tertunda, dan lain-lain). Bahkan ketika suatu proyek menemukan suatu problem atau isu yang tidak terduga, proses dapat disesuaikan dengan berdasarkan pada dampak yang dapat diprediksi dan diukur.

- Level 5 (Optimizing). Pada level ini, proses pengembangan sistem yang sudah distandarisasi dimonitor dan ditingkatkan secara berkelanjutan didasarkan pada ukuran dan analisis data yang dibangun pada level 4. Ini mencakup perubahan teknologi dan best practices yang digunakan untuk menjalankan aktivitas-aktivitas yang diperlukan dalam proses pengembangan sistem standard, sebagaimana penyesuaian proses itu sendiri. Pelajaran yang dapat dipetik disebarkan ke seluruh organisasi, dengan penekanan khusus pada eliminasi ketidakefisienan dalam proses pengembangan sistem dengan tetap mempertahankan kualitas. Secara singkat bahwa peningkatan proses pengembangan sistem yang berkelanjutan telah melembaga dalam organisasi.

Software CMM adalah suatu komponen yang mendukung konsep perbaikan proses yang berkelanjutan. Konsep ini dibangun di SEI Process Improvement IDEAL Model. Model ini digambarkan sebagai berikut :



Phase 1 dari model IDEAL adalah tahap inisiasi (initiation), dimana dukungan manajemen diperoleh untuk memperbaiki proses, sasaran dan kendala dari usaha perbaikan proses yang didefinisikan, dan memperoleh sumberdaya dan rencana untuk tahap selanjutnya.

Phase 2 mengidentifikasi metode penilaian yang sesuai, mengidentifikasi proyek-proyek yang akan dinilai, melatih team penilaian, menjalankan penilaian, melaporkan secara ringkas kepada manajemen dan organisasi mengenai hasil penilaian.

Phase 3, suatu rencana aksi dikembangkan berdasarkan hasil tahap 2, manajemen diberi laporan ringkas mengenai rencana aksi, sumber–sumber daya dan kelompok-kelompok yang dikoordinasikan untuk mengimplementasikan rencana aksi.

Phase 4 adalah tahap aksi, dimana sumber daya direkrut untuk mengimplementasikan rencana aksi, rencana aksi diimplementasikan, usaha-usaha perbaikan diukur, dan rencana dan implementasi dimodifikasi berdasarkan pengukuran dan umpan balik.

Phase 5 adalah tahap review (mengkaji ulang) untuk menjamin bahwa semua kriteria keberhasilan telah dicapai, semua umpan balik dievaluasi, pelajaran yang dipelajari dianalisa, rencana bisnis dan perbaikan proses dibandingkan untuk melihat kesesuaian hasilnya, dan tahap selanjutnya dari usaha perbaikan proses direncanakan.

Manfaat jangka panjang dari rencana perbaikan proses perangkat lunak formal adalah meningkatkan kualitas softwrae, mengurangi waktu siklus hidup, penjadualan dan memenuhi milestones lebih akurat, visibilitas manajemen, dan perencanaan dan pelacakan proaktif

BAB IV. Object-Oriented Systems

Object-oriented system memiliki karakteristik potensial menjadi lebih handal dan mampu mengurangi penyebaran dari program yang berubah menjadi kesalahan, disamping menunjang pemodelan dari dunia nyata. Suatu object-oriented system dapat menjadi pemikiran sebagai kelompok obyek independen yang dapat diminta untuk mengerjakan operasi-operasi tertentu atau menampilkan kelakuan tertentu. Obyek-obyek ini bekerjasama menyediakan fungsionalitas yang dibutuhkan sistem. Obyek-obyek memiliki identitas dan dapat dibuat sebagai program eksekusi. Untuk menyediakan karakteristik yang sesuai dari object-oriented system, obyek-obyek dienkapsulasi yakni mereka hanya dapat diakses melalui message yang dikirimkan kepada mereka untuk meminta performa dari operasi yang mereka definisikan. Obyek dapat dilihat sebagai ‘kotak hitam’ yang detil internalnya tersembunyi dari amatan luar dan tidak dapat dimodifikasi secara biasa. Enkapsulasi didefinisikan Grady Booch sebagai proses penggolongan elemen-elemen dari suatu abstraksi yang merupakan struktur dan kelakuannya. Ia melayani untuk memisahkan antarmuka yang sudah disepakati dari suatu abstraksi dan implementasinya. Obyek-obyek juga menampilkan properti pengganti, yang berarti bahwa obyek menyediakan operasi-operasi yang kompatibel yang dapat disubstitusikan antara satu dengan yang lain. Obyek memiliki status, kelakuan, dan identitas.

Definisi-definisi berikut ini adalah fundamental untuk object-oriented system :

- Message. Message adalah komunikasi ke suatu obyek untuk melaksanakan beberapa opeasi.
- Method. Method adalah kode yang mendefinisikan tindakan yang dilakukan suatu obyek sebagai respon atas suatu message.
- Behavior. Behavior merujuk ke hasil yang diperagakan oleh suatu obyek atas penerimaan suatu message.
- Class. Class adalah suatu kumpulan dari method-method umum dari sekumpulan obyek yang mendefinisikan behavior obyek tersebut.
- Instance. Obyek adalah contoh class yang berisi method-method mereka.

- Inheritance. Method dari suatu class diturunkan oleh subclass lain. Sehingga, subclass menurunkan behavior dari class yang lebih besar, atau sering disebut superclass.
- Multiple Inheritance. Multiple Inheritance adalah situasi dimana suatu class menurunkan karakteristik kelakuannya dari banyak parent class.
- Delegation. Delegation adalah suatu obyek meneruskan suatu permintaan ke obyek lain. Hal ini dilakukan karena suatu obyek menerima suatu permintaan namun tidak memiliki method untuk melayani permintaan tersebut.
- Polymorphism. Suatu nama bisa merupakan obyek-obyek dari beberapa class berbedayang direlasikan oleh superclass. Jadi, beberapa obyek ditunjukkan oleh nama ini yang bisa merespon sejumlah operasi umum dalam cara berbeda.
- Polyinstantiation. Polyinstantiation adalah pengembangan versi terinci suatu obyek dari obyek lain menggunakan nilai berbeda pada obyek baru. Dalam sekuritas informasi database, istilah ini difokuskan pada kunci primer yang sama untuk relasi berbeda pada tingkat klasifikasi berbeda yang disimpan pada database yang sama. Sebagai contoh, dalam suatu database relasional, nama dari suatu unti militer bisa diklasifikasikan secret di database sehingga memiliki suatu nomor identifikasi sebagai kunci primer. Jika pengguna lain pada klasifikasi yang lebih rendah berusaha membuat suatu masukan confidential untuk unit militer lain menggunakan nomor identifikasi yang sama sebagai kunci primer, maka ada penolakan terhadap usaha ini yang secara tidak langsung menunjukkan kepada pengguna pada level dibawahnya bahwa nomor identifikasi yang sama sudah ada pada klasifikasi pada tingkat yang lebih tinggi.

Relatif terhadap tahapan-tahapan pengembangan perangkat lunak, orientasi obyek diterapkan dalam berbagai tahapan sebagai berikut :

a. Object-Oriented Requirement Analysis (OORA). Mendefinisikan class-class obyek dan interaksi mereka.
b. Objcet-Oriented Analysis (OOA). Pada istilah dalam konsep orientasi obyek, pemahaman dan pemodelan permasalahan tertentu didalam suatu domain permasalahan.
c. Domain Analysis (DA). Kekhasan OOA adalah fokus atas satu problem tertentu pada suatu waktu, DA melihat untuk mengidentifikasi class dan obyek yang umum untuk semua aplikasi didalam domain yang diberikan.
d. Object-Oriented Design (OOD). Obyek adalah unit dasar modularitas, dan obyek adalah instantiasi suatu class.
e. Object-Orientef Programming (OOP). Menekankan pekerjaan obyek dan method dari pada jenis atau transformasi sebagaimana dalam pendekatan pemrograman lain.

Contoh sederhana mengenai class adalah class ‘pesawat’. Dari class ini bisa dibuat obyek-obyek seperti pesawat tempur, pesawat angkut, pesawat kargo, pesawat latih. Method yang terkait dengan class ini akan dijalankan ketika obyek menerima suatu message.

Dengan menggunakan obyek yang sudah diuji dan handal, aplikasi dapat dikembangkan dengan waktu yang lebih singkat dan biaya yang lebih sedikit. Obyek-obyek ini dapat dikontrol melalui suatu program library obyek yang mengontrol dan mengelola simpanan dan keluaran obyek-obyek yang sudah diuji untuk pengguna. Untuk melakukan proteksi terhadap penyingkapan dan pelanggaran integritas obyek, maka kontrol sekuritas harus diimplementasikan terhadap program library. Dan lagi, obyek dapat dibuat tersedia bagi user melalui Object Request Broker (ORB). Kegunaan ORB adalah untuk mendukung interaksi antar obyek di lingkungan yang heterogen dan terdistribusi. Obyek mungkin berada pada beraneka jenis platform komputasi. Oleh karena itu, ORB bertindak sebagai penempat dan pendistribusi obyek-obyek melalui jaringan. ORB dipandang sebagai middleware karena mereka berada diantara dua entitas lain. ORB dapat menyediakan fitur sekuritas, atau obyek dapat memanggil layanan sekuritas (security service). Suatu ORB adalah suatu komponen dari Object Request Architecture (ORA), yakni suatu kerangka kerja tingkat tinggi untuk suatu lingkungan terdistribusi. Komponen lain ORA meliputi object service (layanan obyek), application object (obyek aplikasi), dan fasilitas umum.

ORA merupakan produk dari Object Management Group (OMG), suatu konsorsium nonprofit di Framingham, Massachusetts. Kelompok ini dibangun untuk mempromosikan teknologi obyek pada sistem komputasi terdistribusi. Object service mendukung ORA dalam penciptaan dan pelacakan obyek, serta pengerjaan fungsi kendali akses. Application object dan fasilitas umum mendukung pengguna akhir dan menggunakan layanan sistem untuk mengerjakan fungsi mereka.

OMG juga telah mengembangkan Common Object Request Broker Architecture (CORBA), yang telah mendefinisikan standard industri yang memungkinkan program ditulis dalam berbagai macam bahasa dan menggunakan berbagai macam platform dan sistem operasi untuk antar muka dan komunikasi. Untuk mengimplementasikan kompatibilitas antar berbagai bahasa, platform, dan sistem operasi ini, pengguna mengembangkan sejumlah kode inisial dan suatu file Interface Definition language (IDL). File IDL selanjutnya mengidentifikasi method, class, dan object yang merupakan target interface. Contoh, CORBA dapat memungkinkan suatu kode java mengakses dan menggunakan kode C++.

Standard yang lain adalah Common Object Model (COM), mendukung pertukaran object-object antar program. Kemampuan ini sebelumnya dikenal sebagai Object Linking and Embedding (OLE). Sebagaimana model object-oriented, COM bekerja dengan object yang terenkapsulasi (terselubung). Komunikasi dengan object COM adalah melalui suatu kesepakatan interface antara object dengan client-nya yang mendefinisikan fungsi-fungsi yang tersedia pada object dan kelakuan object ketika object dipanggil. Distributed Common Object odel (DCOM) mendefinisikan standard dalam berbagi object di lingkungan jaringan.

Beberapa contoh dari object-oriented system adalah Simula 67, C++, dan Smalltalk. Simula 67 adalah sistem pertama yang mendukung pemrograman berorientasi obyek, namun tidak dipakai secara luas. Tetapi, konstruksinya telah berpengaruh pada

bahasa berorientasi obyek yang lain, termasuk C++. Smalltalk dikembangkan di Xerox Palo Alto Research Center (PARC) sebagai suatu sistem yang utuh.

BAB V. Artificial intelligence systems

Pendekatan lain penggunaan perangkat lunak dan/atau perangkat keras untuk memecahkan permasalahan adalah melalui penggunaan artificial intelligence system. Istem ini berusaha untuk menirukan pekerjaan pikiran manusia. Dua jenis artificial intelligence system yakni expert system dan neural network.

a. Expert System

Expert system memperagakan penalaran yang mirip dengasn seorang ahli dalam memecahkan suatu permasalahan. Ia membangun penalarannya dengan membangun knowledge base dari domain yang akan diselesaikan dalam bentuk aturan-aturan (rule) dan mekanisme penarikan kesimpulan untuk menentukan apakah aturan-aturan telah dipenuhi oleh masukan sistem.

Program komputer biasanya didefinisikan sebagai :

Algorithma + struktur data = program

Dalam expert system, hubungannya adalah

Inference engine + knowledge base = expert system

Knowledge base berisi fakta-fakta dan aturan-aturan yang terkait dengan domain permasalahan dalam bentuk pernyataan if-then. Inference engine membandingkan informasi yang terkumpul di memori dengan bagian if dari aturan yang ada dalam knowledge base untuk melihat apakah ada kesesuaian. Jika ada kesesuaian maka aturan siap ‘eksekusi’ dan ditempatkan dalam daftar eksekusi. Aturan tertentu bisa memiliki prioritas yang lebih tinggi, sehingga sistem akan mengeksekusinya lebih dahulu dari aturan lain yang prioritasnya lebih rendah.

Expert system beroperasi dengan salah satu moda yakni forward-chaining atau backward-chaining. Pada moda forward-chaining, expert system mengumpulkan informasi dan menarik kesimpulan berdasarkan informasi tersebut. Forward-chaining adalah pendekatan penalaran yang dapat digunakan ketika disana ada sejumlah kecil solusi relatif terhadap jumlah input. Sedangkan pada mode backward-chaining, expert system melacak balik untuk menentukan jika hipotesis yang diberikan adalah benar. Backward-chaining secara umum digunakan ketika disana ada sejumlah besar kemungkinan solusi relatif terhadap jumlah input.

Jenis lain expert system adalah blackboard. Blackboard adalah suatu metodologi penalaran sistem ahli dimana solusi dihasilkan melalui penggunaan ‘papan tulis’ virtual, dimana informasi atau solusi potensial ditempatkan pada papan tulis dengan jumlah terbanyak sumber individual atau pengetahuan pakar. Informasi lebih banyak ditempatkan pada papan tulis dalam suatu proses berulang, hingga solusi dihasilkan.

Sebagaimana dengan penalaran manusia, disana ada tingkat ketidakpastian dari kesimpulan yang dihasilkan expert system. Ketidakpastian ini dapat ditangani melalui sejumlah pendekatan seperti bayesian network, certainty factor, atau fuzzy logic.

Bayesian network berlandaskan pada teorema Bayes sebagai berikut :

$$P(H|E) = P(E|H) * P(H) / P(H)$$

Yang memberikan probabilitas kejadian (H) yang diberikan dimana kejadian (E) telah terjadi.

Certainty factors mudah mengembangkannya maupun menggunakannya. Faktor-faktor ini adalah probabilitas yang dipercaya benar. Contoh, probabilitas 85% dapat diberikan ke obyek A terjadi dibawah kondisi tertentu.

Fuzzy logic digunakan untuk menyelesaikan situasi dimana disana ada tingkat ketidakpastian apakah sesuatu benar atau salah. Situasi ini seringkali merupakan situasi di dunia nyata. Suatu fuzzy expert system memadukan fungsi-fungsi fuzzy untuk membangun kesimpulan. Langkah-langkah inference engine pada fuzzy logic adalah sebagai berikut :

Fuzzification. Fungsi-fungsi keanggotaan didefinisikan pada variabel-variabel input dipakai untuk nilai-nilai aktual mereka, untuk menentukan tingkat kebenaran dari setiap aturan yang digunakan.

Inference. Nilai kebenaran untuk setiap aturan dikomputasi dan dipakai untuk kesimpulan bagian dari setiap aturan. Hasil ini dalam satu himpunan bagian fuzzy untuk diberikan ke masing-masing variabel output.

Composition. Seluruh himpunan bagian fuzzy diberikan ke setiap variabel output dikombinasikan bersama membentuk himpunan bagian fuzzy tunggal untuk setiap variabel output.

Defuzzification. Digunakan untuk mengkonversi output fuzzy ke angka kuantitatif. Satu pendekatan defuzzification adalah metoda CENTROID. Dengan metoda ini, nilai dari variabel output dihitung dengan mencari nilai variabel dari pusat situasi fungsi keanggotaan untuk nilai output fuzzy.

Spiral model dapat digunakan untuk membangun suatu expert system. Langkah-langkah umum membangun expert system dengan spiral model mencakup analisis, spesifikasi, pengembangan, dan penyerahan.

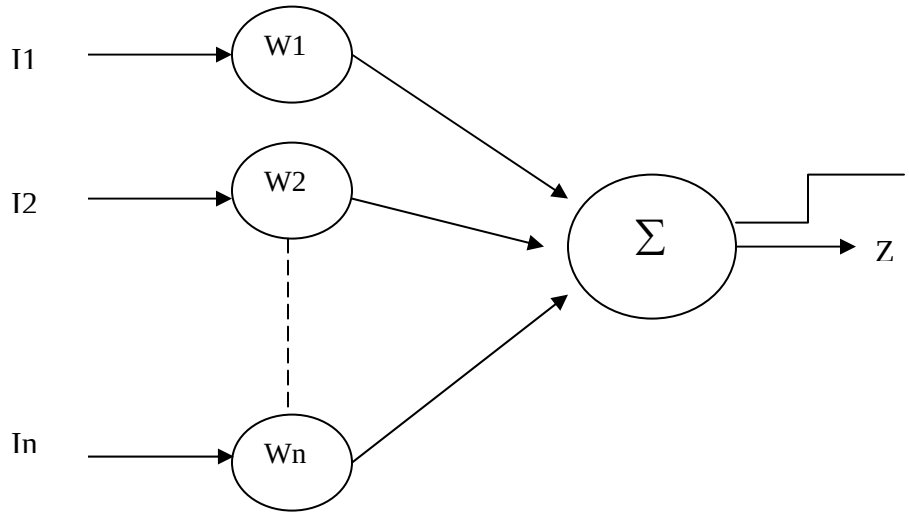
Elemen kunci dalam proses ini adalah penambahan pengetahuan. Aktivitas ini meliputi wawancara dengan ahli sesuai dengan bidang keahliannya dan memperoleh data dari sumber pakar yang lain. Penambahan pengetahuan mulai pada tahap spesifikasi dan berjalan dalam tahap pengembangan.

Verifikasi dan validasi suatu expert system difokuskan pada inkonsistensi yang melekat dalam aturan-aturan yang bertentangan, aturan-aturan yang redundan, rantai aturan yang melingkar, dan nilai yang tidak memiliki referensi bersam-sama dengan ketidaklengkapan hasil yang diperoleh.

b. Neural Network

Neural network berdasarkan fungsi biologi neuron. Dalam biologi neuron, signal dipertukarkan diantara neuron-neuron melalui pulsa elektrik yang berjalan sepanjang axon. Ketika pulsa sampai di synapse, ini menyebabkan pelepasan kimiawi neutrontansmitter yang berjalan melewati synaptic clef hingga post-synaptic receptor yang berada pada sisi dendrite dari synapse. Neutrontransmitter selanjutnya menyebabkan suau perubahan pada dendrite membrane post-synaptic-potential (PSP). PSP ini digabungkan oleh neutrin sepanjang waktu. Jika PSP yang digabungkan melebihi batas, neutron membakar dan menghasilkan suatu pulsa elektrik yang berjalan ke neuron lain.

Suatu analog dengan sistem biologi neuron adalah sebagaimana gambar dibawah ini :



Input Ii ke neuron dimodifikasi oleh Wi, dan dijumlahkan dalam unit ∑. Jika jumlahan Wi melebihi batas, unit ∑ akan menghasilkan output Z. Fungsi dari artificial neural network ditunjukkan dalam persamaan sebagai berikut :

$$Z = W1\ I1 + W2\ I2 + \text{----} + Wn\ In$$

Jika jumlahan dari input W melebihi batas yang ditentukan maka neuron akan membakar dan menjadi output dari neuron tersebut.

Karena disana dalam gambar diatas hanya ada satu node penjumlahan, maka network ini disebut single-layer network. Sedangkan network yang memiliki node penjumlahan lebih dari satu disebut multi-layer network. Nilai dari neural network adalah kemampuannya untuk secara dinamis menyesuaikan W-nya agar menghubungkan vektor input yang diberikan dengan vektor output yang dihasilkan.

Periode ‘pelatihan’untuk neural network memiliki vektor input yang secara berulang ditunjukkan dan Weight (W) secara dinamis diatur sesuai model pembelajaran. Aturan delta adalah contoh aturan pembelajaran. Dalam aturan delta, perubahan weight adalah $\Delta ij = R*Ii * (Tj - Zj)$. Dimana R adalah tingkat pembelajaran, Ii adalah vektor input, dan Tj adalah sasaran vektor output. Sedangkan Zj adalah aktual output dari node ∑. Sebagai contoh, jika suatu vektor aoutput tertentu diperlukan untuk suatu input tertentu dimana hubungan antara input dan output adalah tidak linear, neural network akan di ujicoba dengan penerapan sekumpulan vektor input. Penggunaan aturan delta akan mengatur W secara berulang hingga dihasilkan vektor output yang benar untuk setiap vektor input yang diberikan. Neral network selanjutnya dikatakan telah belajar unuk menyediakan jawaban yang benar untuk setiap vektor input.

BAB VI. Database systems

Suatu sistem database dapat digunakan sebagai mekanisme umum untuk pendefinisian, penyimpanan, dan pemanipulasian data tanpa penulisan program tertentu untuk mengerjakan fungsi ini. Suatu Database Management System (DBMS) menyediakan perintah-perintah tingkat tinggi untuk mengoperasikan data pada database. Ada beberapa jenis database, yakni Hierarchical, Mesh, Object-oriented, dan Relational. Penelitian yang banyak dilakukan terkait dengan sekuritas informasi adalah yang terkait dengan basis data relational.

a. Relational Database

Model basis data relasional memiliki tiga bagian yang meliputi struktur data yang disebut tabel atau relasi, integrity rule yang menentukan nilai atau kombinasi nilai tabel yang diperbolehkan, dan operator-operator yang dapat dioperasikan terhadap tabel.

Suatu basis data dapat didefinisikan sebagai suatu kumpulan dari item-item data saling terkait yang persisten. Persistensi diperoleh melalui pemeliharaan integritasnya dan melalui penggunaan media penyimpanan yang non-volatile. Deskripsi basisdata adalah schema, dan Data Description Language (DDL) mendefinisikan schema. Sedangkan, Suatu Database Management System (DBMS) adalah perangkat lunak yang memelihara dan menyediakan akses ke basisdata. Terkait dengan sekuritas, DBMS dapat diatur bahwa hanya subyek tertentu diijinkan untuk mengerjakan operasi tertentuterhadap basisdata. Sebagai contoh, pengguna tertentu dapat dibatasi untuk informasi tertentu di basisdata dan tidak diperbolehkan untuk melihat beberapa informasi yang lain.

Relasi adalah dasar dari basisdata relasional dan direpresentasikan dalam tabel dua dimensi. Baris tabel merepresentasikan record atau tuple, dan kolom tabel merepresentasikan atribut. Jumlah baris dalam relasi menunjukkan cardinalitas, sedang jumlah kolom menunjukkan degree. Domain dari suatu relasi adalah sekumpulan nilai yang diperbolehkan yang mana atribut dapat mengambilnya. Sebagai contoh relasi adalah Part dan Electricalitem dibawah ini.

Relasi PART

PART NUMBER	**PART NAME**	**PART TYPE**	**LOCATION**
E2C491	Alternator	Electrical	B261
M4D326	Idle Gear	Mechanical	C418
E5G113	Fuel Gauge	Electrical	B561

Relasi ELECTRICAL ITEM

SERIAL NUMBER	**PART NUMBER**	**PART NAME**	**PART COST**
S367790	E2C491	Alternator	$200
S785439	E5D667	Control Module	$700
S677322	E5W459	Wiondow Motor	$300

Pada masing-masing table dibutuhkan primary key (kunci primer). Kunci primer adalah identitas unik dalam tabel yang secara jelas menunjuk ke masing-masing record atau tuple dalam tabel. Suatu kunci primer adalah anggota bagian dari kunci-kunci kandidat (candidate key) di tabel. Suatu kunci kandidat adalah suatu atribut yang merupakan identitas unik dari suatu tabel. Sebagai contoh dalam relasi PART, kunci primer relasi ini adalah Part Number. Apabila location dalam relasi PAERT tersebut, maka ini bisa digunakan sebagai kunci primer. Selanjutnya Part Number dan Location akan dipertimbangkan sebagai kunci-kunci kandidat, dan kunci primer akan diambil dari salah satu dari atribut ini. Sekarang anggap bahwa atribut Part Number sebagai kunci primer dari tabel PART tersebut. Jika suatu atribut dalam satu relasi memiliki nilai sesuai kunci primer di relasi lain, atribut ini disebut foreign Key. Sebagi contoh, atribut Part Number E2C491 pada tabel ELECTRICAL ITEM adalah foreign key karena nilainya terkait ke atribut kunci primer pada tabel PART.

Integritas Entitas dan referensial

Melanjutkan contoh sebelumnya, jika Part Number ditunjuk sebagai kunci primer dari tabel PART, maka selanjutnya setiap baris dari tebel tersebut harus memiliki atribut Part Number. Jika atribut Part Number adalah NULL, maka integritas entitasnya telah dilanggar. Hal serupa, dimana integritas referensial memerlukan bahwa untuk beberapa atribut foreign key, hubungan referensi harus memiliki suatu tuple dengan nilai sama dengan kunci primer. Jadi, jika atribut E2C491 dari tabel ELECTRICAL ITEM adalah foreign key dari tabel PART, maka E2C491 harus menjadi kunci primer di tabel PART untuk memenuhi integritas referensial. Kesesuaian foreign key dengan primary key adalah penting karena mereka merepresentasikan referensi dari satu relasi ke yang lain dan membangun hubungan diantara relasi-relasi ini.

Operasi Basisdata Relasional

Sejumlah operasi dalam aljabar relasional digunakan untuk membangun relasi dan operasi terhadap data. Lima dari operasi-operasi berikut adalah primitif, dan operasi-operasi lain dapat didefinisikan terkait dengan kelima operasi tersebut. Selanjutnya dibahas lebih detil beberapa operasi yang banyak dipakai yaitu :

- Select (primitif)
- Project (primitif)
- Union (primitif)
- Difference (primitif)
- Product (primitif)
- Join
- Intersection
- Divide
- Views

Operasi select mendefinisikan suatu relasi baru berdasarkan pada suatu formula. Sebagai contoh, electrical part yang harganya diatas $300 dari tabel ELECTRICAL ITEM. Operasi join memilih tuple yang memiliki nomor sama untuk beberapa atribut. Contoh, dalam tabel PART dan ELECTRICAL ITEM, Serial Number dan Location digabungkan menggunakan Part Number. Operasi Union membentuk suatu relasi baru dari dua relasi lain. Sebagai gambaran, untuk relasi yang kita sebut X dan Y, relasi baru berisi setiap tuple yang ada dalam salah satu X atau Y atau keduanya.

Suatu operasi penting terkait dengan pengendalian akses informasi basis data adalah view. View didefinisikan dari operasi-operasi Join, Project, dan Select. View tidak ada dalam bentuk fisik, namun dapat dipandang sebagai suatu tabel virtual yang dihasilkan dari tabel lain. Relasi yang benar-benar ada dalam basisdata disebut base relation. Tabel-tabel lain ini dapat berupa tabel yang ada

dalam basisdata atau view yang didefinisikan sebelumnya. View dapat dilihat sebagai suatu cara membangun tabel yang bisa digunakan berulangkali meskipun secara fisik tidak ada dalam basisdata. View dapat digunakan untuk membatasi akses ke informasi tertentu dalam basisdata, untuk menyembunyikan atribut, dan untuk mengimplementasikan pembatasan akses berdasarkan isi. Jadi, seseorang yang minta untuk mengakses informasi dalam basisdata kan diberikan dengan view yang berisi informasi yang diijinkan saja untuk dilihat, dan menyembunyikan informasi lain yang tidak berhak untuk dilihat. Dengan cara ini maka view dapat digunakan untuk mengimplementasikan Least Privilege.

Dalam pengembangan suatu query dari basisdata relasional, suatu optimasi proses dilakukan. Proses ini mencakup pembuatan rencana-rencana query dan memilih yang rencana terbaik, yakni yang biayanya terendah. Query plan tersusun dari implementasi prosedur-prosedur yang terkait ke masing-masing operasi tingkat rendah dari query tersebut. Pemilihan rencana dengan biaya terendah mencakup pemberian biaya untuk rencana tersebut. Biaya bisa menjadi fungsi dari akses disk dan penggunaan CPU.

Dalam query basisdata statistik, mekanisme proteksi yang digunakan untuk membatasi gangguan informasi adalah spesifikasi penentuan ukuran query minimum, pencegahan terhadap permintaan seluruh data, tetapi satu dari record-record dalam databse. Kontrol ini menghalangi suatu penyerangan dari penggabungan statistik suatu query yang diatur brukuran M, sama atau lebih besar dari ukuran minimum query yang ditentukan, dan selanjutnya permintaan ke statistik sama pada quwry yang dipasang dengan ukuran M+1. Sekumpulan query kedua akan dirancang untuk mencakup individu yang informasi sedang dicari diam-diam. Ketika query basisdata untuk informasi statistik, informasi yang dapat diidentifikasi secara individual akan diproteksi. Jadi, keperluan ukuran minimum untuk query set memberikan proteksi terhadap penggabungan informasi pada satu pemakai.

Suatu bind juga dipakai bersama dengan rencana pengembangan query. Bind membuat rencana dan memperbaiki atau merubah rencana itu. Bind variable adalah pemegang tempat (placeholder) untuk nilai-nilai literal dalam Structure Query Language (SQL) dimana query dikirim ke database di server. Statemen SQL dikirim ke server untuk penguraian, selanjutnya nilai diikat ke placeholder dan dikirim secara terpisah ke server.

Normalisasi Data

Normalisasi adalah bagian penting mdalam perancangan basisdata untuk menjamin bahwa atribut-atribut dalam tabel hanya bergantung pada primary key. Proses ini membuat basisdata lebih mudah untuk dipelihara dan memiliki laporan yang konsisten. Normalisasi data terdiri dari tiga langkah yaitu :

1. Menghilangkan kelompok-kelompok yang berulang dengan menempatkan mereka ke dalam tabel yang terpisah.
2. Menghilangkan redundansi data.
3. Menghilangkan atribut-atribut dalam tabel yang tidak bergantung pada primary key tabel tersebut.

SQL

SQL dikembangkan oleh IBM. SQL adalah bahasa standar untuk definisi dan manipulasi data untuk basisdata relasional. Bahasa pendefinisian data SQL membuat dan menghapus view dan relasi (tabel). Perintah-perintah SQL mencakup Select, Update, Delete, Insert, Grant, dan Revoke. Dua perintah terakhir digunakan untuk kontrol akses terhadap hak memberi dan mencabut sumber daya. Biasanya, pemilik suatu obyek dapat menahan atau memberikan hak GRANT yang terkait dengan obyek tersebut kepada subyek lain. Apabila pemilik dengan sengaja tidak memberikan hak GRANT suatu obyek kepada individu A, maka A tidak dapat melanjutkan hak GRANT ke subyek lain. Dalam beberapa hal, kontrol keamanan ini dapat dielakkan. Sebagai contoh, jika A menyalin obyek maka hakekatnya A menjadi pemilik obyek tersebut, selanjutnya dapat memberikan hak GRANT kepada pengguna lain, misalnya B.

Isu-isu Keamanan Basisdata

Dalam basisdata relasional, keamanan dapat disediakan melalui penggunaan *View*. View adalah relasi maya yang mengkombinasikan informasi dari relasi-relasi lain. Suatu View dapat digunakan untuk membatasi data yang tersedia bagi user didasarkan pada hak yang mereka miliki. Kelemahan keamanan informasi basisdata dapat dieksploitasi melalui DBMS. Rancangan untuk memfasilitasi query ke basisdata, DBMS dapat menjadi kemungkinan sumber data sesuai dengan pengelakan kontrol keamanan normal. Granularity akses ke obyek dalam basisdata merujuk ke ‘fineness’ dimana akses ini dapat dikontrol atau dibatasi. Isu keamanan lain terkait dengan basisdata adalah agregasi dan inferensi. Agregasi adalah tindakan untuk mendapatkan informasi dengan sensitivitas yang lebih tinggi dengan cara mengkombinasikan informasi-informasi yang memiliki sensitivitas yang lebih rendah. Inferensi adalah kemampuan pengguna untuk menarik kesimpulan informasi mengenai data-data sensitif yang mereka tidak memiliki hak untuk mengaksesnya. Jalur yang memungkinkan inferensi terjadi disebut inference channel.

Open Database Connectivity (ODBC) adalah standar yang dikembangkan oleh Microsoft untuk mendukung akses ke basisdata melalui berbagai macam aplikasiAkses ini harus dikendalikan untuk menghindari kompromi basisdata.

b. Data Warehouse

Data yang dibutuhkan untuk mendukung pengambilan keputusan berbeda dengan data yang dibutuhkan oleh sistem yang digunakan untuk menjalankan aktifitas bisnis sehari-hari. Kebutuhan data untuk mendukung pengambilan keputusan ini tidak mampu disediakan oleh basisdata yang terdapat dalam sistem operasional. Dalam pengambilan keputusan memerlukan data-data yang bisa digunakan untuk analisa, melihat kecenderungan, dan memonitor performa. Untuk itu diperlukan lingkungan sistem baru yang mampu menyediakan informasi strategis.

Karakteristik lingkungan sistem yang sesuai untuk mendukung ketersediaan data dalam pengambilan keputusan adalah sebagai berikut :

- Basisdata dirancang untuk membantu kegiatan analisa.
- Data bersumber dari banyak aplikasi.
- Mudah digunakan dan kondusif untuk menjalankan interaksi yang panjang dari user.
- Penggunaan data lebih mendukung untuk proses pembacaan data yang intensif.
- Pengguna tanpa bantuan dapat berinteraksi langsung dengan sistem.
- Isi data dimutakhirkan secara periodik dan stabil.
- Isi data mencakup data saat ini maupun data historis.
- Sistem memberikan kemampuan bagi pengguna untuk menjalankan query dan mendapatkan hasilnya segera.
- Sistem memberikan kemampuan bagi pengguna untuk menginisiasi laporan.

Organisasi yang sedang membangun data warehouse, sebenarnya sedang membangun lingkungan baru dengan karakteristik seperti diuraikan diatas. Lingkungan baru ini terpisah dari lingkungan sistem yang mendukung operasi sehari-hari. Data warehouse intinya adalah kemampuannya untuk menyediakan informasi strategis untuk mendukung pengambilan keputusan. Sumber data dari data warehouse diperoleh dari sistem-sistem operasional yang mendukung proses bisnis dasar organisasi. Antara sistem operasional dan data warehouse dihubungkan oleh ‘staging area’. Pada staging area ini, data-data operasional dibersihkan dan ditransformasikan kedalam bentuk yang sesuai dengan format data warehouse.

Konsep data warehouse adalah sederhana, bukan untuk menghasilkan data baru, tetapi untuk memanfaatkan besarnya data yang sudah dimiliki dan mentransformasikannya ke bentuk yang sesuai untuk penyediaan informasi strategis. Keberadaan data warehouse adalah untuk menjawab pertanyaan-pertanyaan pengguna tentang bisnis, tentang performa dari berbagai operasi bisnis, tentang kecenderungan bisnis, dan tentang apa yang dapat dilakukan untuk meningkatkan bisnis. Data warehouse ada untuk menyediakan pengguna melakukan akses langsung ke data, menyediakan bagi pengguna kemampuan untuk melihat data dari berbagai sudut pandang. Hal ini dilakukan oleh data warehouse dengan konsep yang sederhana, yakni ambil seluruh data yang dimiliki organisasi, bersihkan dan transformasikan data tersebut, dan selanjutnya menyediakan informasi strategis yang diperlukan oleh pengguna.

Data warehouse bukan suatu produk perangkat lunak atau perangkat keras tunggal yang dibeli untuk menyediakan informasi strategis. Namun, ini adalah suatu lingkungan komputasi dimana pengguna dapat memperoleh informasi strategis. Suatu lingkungan dimana pengguna ditempatkan secara langsung berhubungan dengan data yang mereka butuhkan untuk membuat keputusan dengan lebih baik. Kegiatan-kegiatan dalam lingkungan ini secara lebih detil dapat diuraikan sebagai berikut :

- Mengambil seluruh data dari sistem operasional
- Jika perlu, mengambil data-data yang relevan dari lingkungan luar.
- Mengintegrasikan seluruh data dari berbagai sumber tersebut.
- Menghilangkan data-data yang tidak konsistensi dan mentransformasikannya.
- Menyimpan data kedalam format yang mudah diakses untuk pengambilan keputusan.

Meskipun tampaknya ini merupakan suatu konsep sederhana, namun ini mencakup beberapa fungsi yang berbeda-beda. Fungsi-fungsi ini meliputi ekstraksi data, pemuatan data, transformasi data, penyimpanan data, dan penyediaan bagi pengguna. Teknologi-teknologi yang berbeda-beda dibutuhkan untuk mendukung fungsi-fungsi ini. Data warehouse merupakan perpaduan dari berbagai macam teknologi yang diperlukan untuk berbagai macam fungsi. Meskipun berbagai macam teknologi ini digunakan, mereka semua bekerja bersma-sama dalam suatu data warehouse. Hasil akhir adalah penciptaan lingkuangan baru untuk kegunaan penyediaan informasi strategis.

c. Data Mining

Ada berbagai variasi dari definisi data mining. Beberapa ahli memasukkan cakupan menyeluruh mengenai tool dan teknik dalam pendefinisian data mining, dari mekanisme query sederhana hingga analisa statistik. Yang lain membatasi definisi data mining sebagai teknik mencari pengetahuan.

Data mining digunakan disetiap area bisnis dari penjualan dan pemasaran, pengembangan produk baru, manajemen inventori hingga manjemen sumber daya manusia. Berikut beberapa alasan mendasar banyak digunakan dalam bisnis :

- Pada saat ini, suatu organisasi menghasilkan informasi dalam seminggu lebih banyak dari kebanyakan orang dapat membaca dalam seumur hidup. Ini tidak memungkinkan orang untuk mempelajari, memahami, dan menginterpretasikan seluruh data untuk menemukan informasi yang berguna.
- Suatu data warehouse mengelompokkan data setelah pembersihan dan transformasi kedalam struktur data yang terorganisir baik. Namun demikian, volume data saja tidak memungkinkan bagi seseorang untuk mengunakan tool query dan analisa untuk melihat pola-pola yang bermanfaat.
- Data mining membutuhkan kekuatan komputasi yang substansial. Parallel hardware, basis data, dan komponen-komponen yang kuat menjadi sangat terjangkau.

- Organisasi memberi prioritas yang kuat untuk mendengar bagaimana hubungan denagn konsumen. Perusahaan ingin mengetahui bagaimana dapat menjual lebih banyak kepada konsumen yang ada. Organisasi tertarik untuk mengetahui konsumen yang mana akan menjamin keberadaan nilai jangka panjang bagi mereka. Organisasi perlu menemukan klasifikasi dari konsumen sehingga target produk dan layanan klasifikasi bisa dilakukan secara tepat.

Data mining merupakan pendekatan untuk memperoleh informasi yang digerakan oleh data (data-driven), berbeda dengan sistem basis data dan data warehouse dimana pencarian informasi digerakan oleh pengguna (user-driven). Dengan pendekatan user-driven, besarnya data yang tersedia tidak mungkin bagi seseorang dengan menggunakan tool analisa dan query untuk melihat pola-pola yang berguna. Sebagai contoh dalam analisa pemasaran, adalah hampir tidak mungkin memikirkan dari awal hingga akhir semua asosiasi yang dimungkinkan dan mendapatkan pemahaman yang mendalam dengan query dan drill down pada data warehouse. Ini memerlukan teknologi yang dapat belajar dari asosiasi-asosiai sebelumnya serta hasilnya, dan memprediksi kelakuan konsumen. Ini memerlukan tool yang dapat melakukan penemuan pengetahuan knowledge) oleh dirinya sendiri. Jadi, dalam hal ini memerlukan pendekatan data-driven, bukan user-driven.

Jika penemuan pengetahuan adalah satu aspek dari data mining, maka prediksi adalah aspek yang lain. Disini mencari asosiasi tertentu dengan memperhatikan suatu kejadian atau kondisi. Dalam suatu perusahaan, konsumen mereka kemungkinan besar akan menambah jumlah pembalian jika diberi promosi secara tepat. Perusahaan ingin mengetahui tendesi yang mendorong mereka mau mengeluarkan uang lebih banyak untuk berbelanja. Data konsumen bisa berisi asosiasi menarik antara tendensi meningkatkan pembelian dengan umur, tingkat pendapatan, dan status perkawinan. Perusahaan ingin mengetahui faktor-faktor yang mengarahkan tendensi konsumen dan prediksi konsumen yang mana yangv kemungkinan besar meningkatkan pembelian mereka. Disini, prediksi adalah aspek lain dari data mining.

Jadi, data mining adalah proses menemukan pengetahuan. Data mining membantu memahami isi suatu data dengan cara tertentu yang tidak biasa. Ini menaggali pola-pola dan kecenderungan-kecenderungan yang terdapat dalam baris-baris data yang tidak pernah diketahui keberadaannya. Data mining berkisar pada penemuan yang diotomasi mengenai fakta-fakta baru dan hubungan-hubungannya dalam data. Jika tool-tool query tradisional adalah untuk mencari informasi yang ingin diketahui, maka tool data mining memungkinkan penemuan informasi tersembunyi yang tidak muncul.

BAB VII. Usaha Kecil Menengah (UKM) dan Pengembangan Sistem

Dalam bab ini akan dibahas bagaimana implementasi pengembangan sistem dalam suatu UKM. UKM yang akan dijadikan dalam obyek pembahasan disini adalah suatu Mini Market yang dibangun oleh suatu induk koperasi suatu organisasi dengan tujuan melayani kebutuhan anggota organisasi. Adapun jumlah anggota organisasi

sebanyak kurang lebih 2000 orang, dimana jumlah ini hampir tidak berubah dari tahun ke tahun. Sedangkan barang yang dijual oleh Mini Market ini ragam barangnya hampir sama, namun ada peningkatan walaupun tidak banyak dari waktu ke waktu.

Diasumsikan bahwa manajemen menentukan kebijakan bahwa akan dibangun sistem informasi untuk meningkatkan pelayanan pada konsumen dan meningkatkan efektivitas dan efisiensi proses bisnis yang berlangsung dalam organisasi. Sistem informasi akan dikembangkan secara internal, walaupun jumlah personel dibidang ini terbatas.

Dengan mempertimbangkan bahwa cakupan fungsi-fungsi yang akan dibangun tidak begitu luas dan pasti, maka model proses yang akan digunakan adalah waterfall. Sedangkan sistem basis data yang digunakan adalah basis data relasional yang tentu saja juga dilakukan normalisasi. Data warehouse dan data mining belum mendesak dibangun. namun kalaupun dibangun nantinya maka data warehouse dan data mining terbatas, misalnya, untuk mengetahui barang apa yang paling laku terjual atau bagaimana penempatan suatu barang agar penjualan meningkat. Adapun expert system tidak perlu dipergunakan dalam sistem ini.

Seperti pengembangan sistem pada umumnya, ukuran keberhasilan pengembangan sistem adalah bahwa sistem berhasil dibangun tepat waktu, tepat biaya, dan terpenuhinya kebutuhan pengguna. Terkait dengan hal ini maka disiplin dari setiap stakeholder mengikuti prosedur yang disepakati menjadi aspek yang penting.

Referensi

1. Ronald L. Krutz dan Russel D. Vines, “ The CISSP Prep Guide : Gold Edition “, John Wiley & Sons, 2003.
2. J.L. Whitten, L.D. Bentley, dan K.C. Dittman, “ Systems Analysis and Design Methods –6th ed.“, The McGraw-Hill, 2004.
3. Jim Conallen, “ Building Web Applications with UML – 2nd ed. “, Pearson Education, 2003.
4. Paulraj Ponniah, “ DATA WAREHOUSING FUNDAMENTALS : A Comprehensive Guide for IT Professionals “, John Wiley & Sons, 2001.
5. Roger S. Pressman, Ph.D. “ SOFTWARE ENGINEERING : A Practitioner’s Approach – 6th ed. “, The McGraw-Hill Companies, 2005.