BAB II

LANDASAN TEORI

2.1 Pengertian Sistem

Sistem adalah kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu.

1.1.1 Karakteristik Sistem

Karakteristik Sistem / Elemen Sistem :

- **Memiliki komponen ;**

  Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, bekerja sama membentuk satu kesatuan. Komponen-komponen sistem dapat berupa suatu subsistem atau bagian-bagian dari sistem. Setiap sistem tidak perduli betapapun kecilnya, selalu mengandung komponen-komponen atau subsistem-subsistem. Setiap subsistem mempunyai sifat-sifat dari sistem untuk menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan. Suatu sistem dapat mempunyai suatu sistem yang lebih besar yang disebut supra sistem, misalnya suatu

perusahaan dapat disebut dengan suatu sistem dan industri yang merupakan sistem yang lebih besar dapat disebut dengan supra sistem. Kalau dipandang industri sebagai suatu sistem, maka perusahaan dapat disebut sebagai subsistem. Demikian juga bila perusahaan dipandang sebagai suatu sistem, maka sistem akuntansi adalah subsistemnya.

- **Batas sistem (boundary) ;**

  Batas sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batas sistem ini memungkinkan suatu sistem dipandang sebagai suatu kesatuan. Batas suatu sistem menunjukkan ruang lingkup (scope) dari sistem tersebut.

- **Lingkungan luar sistem (environment) ;**

  Adalah apapun di luar batas dari sistem yang mempengaruhi operasi sistem.

- **Penghubung sistem (interface) ;**

  Merupakan media penghubung antara satu subsistem dengan subsistem yang lainnya.

- **Masukan sistem (input) ;**

  Merupakan energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan (maintenance

input) dan masukan sinyal (signal input). Maintenance input adalah energi yang dimasukkan supaya sistem tersebut dapat beroperasi. Signal input adalah energi yang diproses untuk didapatkan keluaran. Sebagai contoh didalam sistem komputer, program adalah maintanance input yang digunakan untuk mengoperasikan komputernya dan data adalah signal input untuk diolah menjadi informasi.

- **Keluaran sistem (Output) ;**

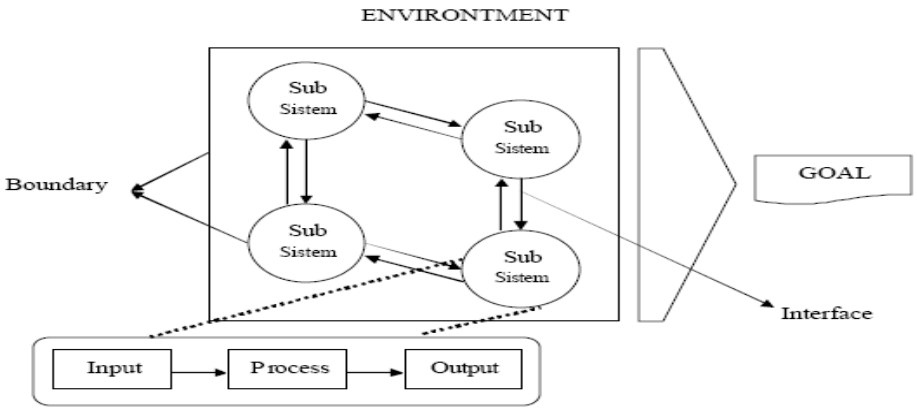
  Merupakan hasil dari energi yang diolah oleh sistem.

- **Pengolah sistem (Process) ;**

  Merupakan bagian yang memproses masukan untuk menjadi keluaran yang diinginkan.

- **Sasaran sistem ;**

  Kalau sistem tidak mempunyai sasaran, maka operasi sistem tidak akan ada gunanya



Gambar Karakteristik Suatu Sistem

Gambar 2.1 Karakteristik Suatu Sistem.

1.1.2 Klasifikasi Sistem

Klasifikasi Sistem :

- **Sistem abstrak** ; sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik (sistem teologia)

  Sistem fisik ; merupakan sistem yang ada secara fisik (sistem komputer, sistem akuntansi, sistem produksi dll.)
- **Sistem alamiah** ; sistem yang terjadi melalui proses alam. (sistem matahari, sistem luar angkasa, sistem reproduksi dll.

  Sistem buatan manusia ; sistem yang dirancang oleh manusia. Sistem buatan manusia yang melibatkan interaksi manusia dengan mesin disebut human-machine system (contoh ; sistem informasi)
- **Sistem Tertentu (deterministic system)** ; beroperasi dengan tingkah laku yang sudah dapat diprediksi. Interaksi bagian-bagiannya dapat dideteksi dengan pasti sehingga keluaran dari sistem dapat diramalkan (contoh ; sistem komputer)

Sistem tak tentu (probabilistic system) ; sistem yang kondisi masa depannya tidak dapat diprediksi karena mengandung unsur probabilitas.

- **Sistem tertutup (close system)** ; sistem yang tidak berhubungan dan tidak terpengaruh dengan sistem luarnya. Sistem ini bekerja secara otomatis tanpa adanya turut campur tangan dari pihak luarnya. Secara teoritis sistem tersebut ada, tetapi kenyataannya tidak ada sistem yang benar-benar tertutup, yang ada hanyalah relatively closed system (secara relatif tertutup, tidak benar-benar tertutup).
- **Sistem terbuka (open system)** ; sistem yang berhubungan dan terpengaruh dengan lingkungan luarnya.
- **Sistem sederhana dan Sistem kompleks**

1.2 Pengertian Informasi

Informasi adalah data yang telah diproses menjadi bentuk yang memiliki arti bagi penerima dan dapat berupa fakta, suatu nilai yang bermanfaat. Jadi ada suatu proses transformasi data menjadi suatu informasi == input - proses – output.

Data merupakan raw material untuk suatu informasi. Perbedaan informasi dan data sangat relatif tergantung pada nilai gunanya bagi manajemen yang memerlukan. Suatu informasi bagi level manajemen tertentu bisa menjadi data bagi manajemen level di atasnya, atau sebaliknya.

Representasi informasi: pelambangan informasi, misalnya: representasi biner.

Kuantitas informasi***:*** satuan ukuran informasi. Tergantung representasi. Untuk representasi biner satuannya: bit, byte, word dll.

Kualitas informasi: bias terhadap error, karena: kesalahan cara pengukuran dan pengumpulan, kegagalan mengikuti prosedur prmrosesan, kehilangan atau data tidak terproses, kesalahan perekaman atau koreksi data, kesalahan file histori/master, kesalahan prosedur pemrosesan ketidak berfungsian sistem.

Umur informasi: kapan atau sampai kapan sebuah informasi memiliki nilai/arti bagi penggunanya. Ada *condition informasion* (mengacu pada titik waktu tertentu) dan *operating information* (menyatakan suatu perubahan pada suatu range waktu)

Kualitas Informasi ; tergantung dari 3 hal, yaitu informasi harus :

- Akurat, berarti informasi harus bebas dari kesalahan-kesalahan dan tidak bias atau menyesatkan. Akurat juga berarti informasi harus jelas mencerminkan masudnya.
- Tetap pada waktunya, berarti informasi yang datang pada penerima tidak boleh terlambat.
- Relevan, berarti informasi tersebut menpunyai manfaat untuk pemakainya. Relevansi informasi untuk tiap-tiap orang satu dengan yang lainnya berbeda.

Nilai Informasi ; ditentukan dari dua hal, yaitu manfaat dan biaya mendapatkannya. Suatu informasi dikatakan bernilai bila manfaatnya lebih efektif

dibandingkan dengan biaya mendapatkannya. Pengukuran nilai informasi biasanya dihubungkan dengan analisis cost effectiveness atau cost benefit.

1.3 Pengertian Sistem Informasi

Sistem Informasi adalah suatu sistem terintegrasi yang mampu menyediakan informasi yang bermanfaat bagi penggunanya. Atau Sebuah sistem terintegrasi atau sistem manusia-mesin, untuk menyediakan informasi untuk mendukung operasi, manajemen dalam suatu organisasi.

Sistem ini memanfaatkan perangkat keras dan perangkat lunak komputer, prosedur manual, model manajemen dan basis data.

Dari definisi di atas terdapat beberapa kata kunci :

1. Berbasis komputer dan Sistem Manusia/Mesin
   - Berbasis komputer: perancang harus memahami **pengetahuan komputer** dan **pemrosesan informasi**
   - Sistem manusia mesin: ada **interaksi** antara manusia sebagai pengelola dan mesin sebagai alat untuk memroses informasi. Ada proses manual yang harus dilakukan manusia dan ada proses yang terotomasi oleh mesin. Oleh karena itu diperlukan suatu prosedur/manual sistem.
2. Sistem basis data terintegrasi

- Adanya penggunaan basis data secara bersama-sama (sharing) dalam sebuah *data base manajemen system*.

3. Mendukung Operasi
    - Informasi yang diolah dan di hasilkan digunakan untuk mendukung operasi organisasi.

Istilah Sistem Informasi

- Manajemen Information System
- Information Processing System
- Information Decision System
- Information System.

Semuanya mengacu pada sebuah sistem informasi berbasis komputer yang dirancang untuk mendukung operasi, manajemen dan fungsi pengambilan keputusan suatu organisasi.

Menurut Robert A. Leitch ; sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan.

Komponen Fisik Sistem Informasi:

1. *Perangkat keras komputer*: CPU, Storage, perangkat Input/Output, Terminal untuk interaksi, Media komunikasi data
2. *Perangkat lunak komputer*: perangkat lunak sistem (sistem operasi dan utilitinya), perangkat lunak umum aplikasi (bahasa pemrograman), perangkat lunak aplikasi (aplikasi akuntansi dll).
3. *Basis data*: penyimpanan data pada media penyimpan komputer.
4. *Prosedur*: langkah-langkah penggunaan sistem
5. *Personil* untuk pengelolaan operasi (SDM), meliputi:
   - *Clerical personnel* (untuk menangani transaksi dan pemrosesan data dan melakukan inquiry = operator);
   - *First level manager*: untuk mengelola pemrosesan data didukung dengan perencanaan, penjadwalan, identifikasi situasi out-of-control dan pengambilan keputusan level menengah ke bawah.
   - *Staff specialist*: digunakan untuk analisis untuk perencanaan dan pelaporan.
   - *Management*: untuk pembuatan laporan berkala, permintaan khsus, analisis khusus, laporan khsusus, pendukung identifikasi masalah dan peluang.

Aplikasi = program + prosedur pengoperasian.

1.4 Metode Pendekatan dan Pengembangan Sistem

1.4.1 Metode Pendekatan Sistem

Tiga jenis usaha pendekatan sistematis untuk pecahan masalah :

1. Persiapan

   Manajer memandang perusahaan sebagai suatu sistem dengan memahami lingkungan perusahaan dan mengidentifikasi subsistem-subsistem dalam perusahaan.

2. Definisi

   Manajer bergerak dari tingkat sistem ke subsistem dan menganalisis bagian sistem menurut suatu urutan tertentu.

3. Solusi

   Manajer mengidentifikasi berbagai solusi altenatif, mengevaluasi, memilih yang terbaik, menerapkannya dan membuat tindak lanjut utk memastikan bahwa solusi itu berjalan sebagaimana mestinya.

PEMAHAMAN DASAR PEMECAHAN MASALAH DAN PEMBUATAN KEPUTUSAN

- Masalah adalah suatu kondisi yg memiliki potensi utk menimbulkan kerugian luar biasa atau menghasilkan keuntungan luar bisa.
- Jadi pemecahan masalah berarti tindakan memberi respon terhadap masalah untuk menekan akibat buruknya atau memanfaatkan peluang keuntungannya.
- Pentingnya pemecahan masalah bukan didasarkan pada jumlah waktu yang dihabiskan tetapi pada konsekuensinya.
- Keputusan adalah pemilihan suatu strategi atau tindakan.
- Pengambilan keputusan adl tindakan memilih strategi atau aksi yg manajer yakini akan memberikan solusi terbaik atas masalah tsb.
- Salah satu kunci pemecahan masalah adalah identifikasi berbagai alternatif keputusan.
- Setelah berbagai alternatif diidentifikasi, sistem informasi dapat digunakan untuk mengevaluasi tiap alternatif.

Evaluasi ini harus mempertimbangkan berbagai **kendala**

1. **Kendala intern** dapat berupa SD yg terbatas, seperti kurangnya bahan baku, modal kerja, SDM yg kurang memenuhi syarat, dll.
2. **Kendala lingkungan** dapat berupa tekanan dari berbagai elemen lingkungan, seperti pemerintah atau pesaing untuk bertindak menurut cara tertentu.

STRUKTUR MASALAH

1. **Masalah terstruktur** terdiri elemen-elemen dan hubungan-hubungan antar elemen yang semuanya dipahami oleh pemecah masalah.
2. **Masalah tak terstruktur** berisikan elemen-elemen atau hubungan-hubungan antar elemen yang tidak dipahami oleh pemecah masalah.
3. **Masalah semi-terstruktur** adalah masalah yang berisi sebagian elemen – elemen atau hubungan yang dimengerti oleh pemecah masalah.

PENDEKATAN SISTEM

- Proses pemecahan masalah secara sistematis bermula dari John Dewey yang mengidentifikasi 3 seri penilaian yang

terlibat dalam memecahkan masalah suatu kontroversi secara memadai yaitu:

1. Mengenali kontroversi
2. Menimbang klaim alternatif
3. Membentuk penilaian

- Kerangka kerja yang dianjurkan untuk penggunaan komputer dikenal sebagai **pendekatan sistem .** Serangkaian langkah-langkah pemecahan masalah yang memastikan bahwa masalah itu pertama-tama dipahami, solusi alternatif dipertimbangkan, dan solusi yang dipilih bekerja.

TAHAPAN PEMECAHAN MASALAH DENGAN MENGGUNAKAN PENDEKATAN SISTEM

- **Usaha persiapan,** mempersiapkan manajer untuk memecahkan masalah dengan menyediakan orientasi sistem.

- **Usaha definisi,** mencakup mengidentifikasikan masalah untuk dipecahkan dan kemudian memahaminya.

- **Usaha solusi,** mencakup mengidentifikasikan berbagai solusi alternatif, mengevaluasinya, memilih salah satu yang tampaknya terbaik, menerapkan solusi itu dan membuat

tindak lanjutnya untuk menyakinkan bahwa masalah itu terpecahkan.

CBIS dapat digunakan sebagai sistem dukungan (*support systems*) saat menerapkan pendekatan sistem.

1. Usaha persiapan

3 langkah persiapan tidak harus dilaksanakan secara berurutan, karena ketiganya bersama-sama menghasilkan kerangka pikir yang diinginkan untuk mengenai masalah.

a) Memandang perusahaan sebagai suatu sistem

b) Mengenal sistem lingkungan

c) Mengidentifikasikan subsistem-subsistem perusahaan

2. Usaha definisi

Usaha definisi mencakup pertama-tama menyadari bahwa suatu masalah ada atau akan ada (identifikasi masalah) dan kemudian cukup mempelajarinya utk mencari solusi (pemahaman masalah)

a) Bergerak dari tingkat sistem ke subsistem

b) Menganalisis bagian sistem dalam suatu urutan tertentu

3. Usaha pemecahan

Usaha pemecahan meliputi pertimbangan berbagai alternatif yang layak (*feasible*), pemilihan alternatif terbaik, dan penerapannya.

2.4.2 Metode Pengembangan Sistem

Pengembangan Sistem

Pengembangan Sistem dapat berarti menyusun suatu sistem yg baru untuk menggantikan sistem yg lama secara keseluruhan atau memperbaiki sistem yg telah ada.

Sebab Perlunya pengembangan Sistem :

- Adanya permasalahan (problems) yg timbul pada sistem yg lama._Permasalahan yg timbul dapat berupa :

o Ketidakberesan

Yang menyebabkan sistem lama tidak beroperasi sesuai dengan yang diharapkan.

Ketidakberesan ini dapat berupa :

- kecurangan yg disengaja yg menyebabkan tdk amannya harta
- kesalahan yg tidak disengaja

- tidak efisiennya operasi
- tidak ditaatinya kebijaksanaan manajemen yang berlaku

- Pertumbuhan Organisasi

- Untuk meraih kesempatan (opportunities) Teknologi informasi telah berkembang dengan cepatnya.
- Adanya instruksi-instruksi (directives)

Prinsip Pengembangan Sistem :

- Sistem yang dikembangkan adalah untuk manajemen.
- Sistem yang dikembangkan adalah investasi modal yang besar.

Setiap investasi modal harus mempertimbangkan 2 hal berikut ini :

 Semua alternatif yang ada harus diinvestigasi

Investor harus memeriksa semua alternatif yang ada dengan melihat opportunity cost dari masing-masing alternative

 Investasi yang terbaik harus bernilai

manfaat (benefit) atau hasil baliknya harus lebih besar dari biaya untuk memperolehnya(cost). Cost-benefit analysis dapat digunakan untuk menentukan apakah proyek investasi tsb bernilai atau tidak.

Sistem yang dikembangkan memerlukan orang yang terdidik.

Seperti Analis sistem, Manajer sistem dan programmer, serta user yang dididik dengan di berikan on-the-job training.

Tahapan kerja dan tugas yang harus dilakukan dalam proses pengembangan system.

Proses pengembangan sistem umumnya melibatkan beberapa tahapan kerja & melibatkan beberapa personil dalam bentuk suatu *team* untuk menjalankannya. Siklus pengembangan Sistem (System Development Life Cycle (SDLC)) umumnya menunjukkan tahap – tahap kerja yg harus dilakukan.

- Proses Pengembangan Sistem tidak harus urut
- Jangan Takut membatalkan proyek
- Dokumentasi harus ada untuk pedoman dalam pengembangan system

Tahapan Utama Siklus hidup Pengembangan Sistem terdiri dari:

- Perencanaan Sistem (systems planning)
- Analisis Sistem (systems analysis)
- Perancangan Sistem (systems design)

- Seleksi Sistem (systems selection)
- Implementasi & pemeliharaan sistem (system implementation & maintenance)

Tahapan - tahapan diatas sebenarnya merupakan tahapan didalam pengembangan sistem teknik (engineering systems).

Terdapat beberapa pendekatan untuk mengembangkan sistem yaitu :

Dipandang dari metodologi yang digunakan :

 Pendekatan Klasik (Clasical approach)

Disebut juga pengembangan tradisional / konvensional adalah pengembangan sistem dengan mengikuti tahapan pada system life cycle. Pendekatan ini menekankan bahwa pengembangan sistem akan berhasil bila mengikuti tahapan pada system life cycle. Tetapi pada kenyataannya pendekatan klasik tidak cukup digunakan untuk mengembangkan suatu sistem informasi yang sukses dan akan timbul beberapa permasalahan diantaranya adalah :

- Pengembangan perangkat lunak menjadi sulit.
- Biaya perawatan atau pemeliharaan sistem menjadi lebih mahal
- Kemungkinan kesalahan sistem besar
- Keberhasilan sistem kurang terjamin

- Masalah dalam penerapan sistem

 Pendekatan Terstruktur (structured approach)

Pendekatan ini dimulai pada awal tahun 1970, dan dilengkapi dengan alat-alat (tools) dan teknik-teknik (techniques) yg dibutuhkan dalam pengembangan sistem.

Dipandang dari sasaran yang dicapai :

 Pendekatan Sepotong (piecerneal approach)

Pendekatan yg menekankan pada suatu kegiatan / aplikasi tertentu.

 Pendekatan Sistem (systems approach)

Pendekatan yg menekankan pada sistem informasi sebagai satu kesatuan terintegrasi

Dipandang dari cara menentukan kebutuhan dari Sistem :

 Pendekatan Bawah Naik (Bottom Up Approach)

Pendekatan dari level bawah organisasi, yaitu level operasional dimana transaksi dilakukan. Pendekatan ini dimulai dari perumusan kebutuhan untuk menangani transaksi dan naik ke level atas dengan merumuskan kebutuhan informasi berdasarkan transaksi tsb. (merupakan ciri-ciri dari pendekatan klasik disebut juga data analysis) .

 Pendekatan Atas Turun

Dimulai dari level atas yaitu level perencanaan strategi. Pendekatan ini dimulai dengan mendefinisikan sarasan dan kebijaksanaan organisasi , kemudian dilakukan analisis kebutuhan informasi , lalu proses turun ke pemrosesan transaksi (merupakan ciri-ciri dari pendekatan terstruktur disebut juga decision analysis)

Dipandang dari cara mengembangkannya :

- Pendekatan Sistem menyeluruh

Pendekatan yg mengembangkan sistem serentak secara menyeluruh.

(merupakan ciri -ciri pendekatan klasik)

- Pendekatan Moduler

Pendekatan yg berusaha memecah sistem yg rumit menjadi beberapa bagian / modul yg sederhana (merupakan ciri -ciri pendekatan terstruktur)

Dipandang dari teknologi yg digunakan :

- Pendekatan Lompatan jauh (great loop approach)

Pendekatan yg menerapkan perubahan menyeluruh secara serentak penggunaan teknologi canggih. Perubahan ini banyak mengandung resiko, juga memerlukan investasi yg besar.

- Pendekatan Berkembang (evolutionary approach)

Pendekatan yg menerapkan perubahan canggih hanya untuk aplikasi yg memerlukan saja, dan akan terus berkembang.

ALAT & TEKNIK PENGEMBANGAN SISTEM

Alat-alat Pengembangan sistem yg berbentuk grafik diantaranya :

- HIPO Diagram digunakan di metodologi HIPO
- Data Flow Diagram digunakan di metodologi structured systems analysis and design
- Structured Chart digunakan di metodologi structured systems analysis and design
- SADT Diagram digunakan di metodologi SADT
- Warnier/Orr Diagram digunakan di metodologi Warnier/Orr
- Jakson's Diagram digunakan di metodologi JSD (Jackson System Development)

Disamping Alat-alat Pengembangan sistem berbentuk grafik yg digunakan pada suatu metodologi tertentu, masih terdapat beberapa alat berbentuk grafik yg sifatnya umum, alat-alat ini berupa suatu bagan.

Bagan dapat diklasifikasikan sebagai berikut :

- Bagan untuk menggambarkan aktivitas (activity charting)

- Bagan alir sistem (systems flowchart)
- Bagan alir program (program flowchart) yang dapat berupa :

  - Bagan alir logika program (program logic flowchat)

  - Bagan alir program komputer terinci (detailed computer program flowchart)
- Bagan alir kertas kerja (paperwork flowchart) atau disebut bagan alir formulir (form flowchart)
- Bagan alir hubungan database (database relationship flowchart)
- Bagan alir proses (process flowchart)
- Gantt Chart
- Bagan untuk menggambarkan tataletak (layout charting)
- Bagan untuk menggambarkan hubungan personil (personal relationship charting)
- Bagan distribusi kerja (Working distribution chart)
- Bagan Organisasi (Organization Chart)

Teknik yg tersedia untuk pengembangan sistem biasanya tidak khusus untuk suatu metodologi tertentu , tetapi dapat digunakan disemua metodologi yg ada. Teknik -teknik yg digunakan :

- Teknik Manajemen Proyek yaitu :
  - CPM (Critical Path Method)

- PERT (Program Evaluation and Review Technique)

Teknik ini digunakan untuk penjadualan proyek

- Teknik menemukan fakta (fact finding techniques)

  Teknik yg dapat digunakan utk mengumpulkan data & menemukan fakta dalam kegiatan mempelajari sistem yg ada.

  Teknik ini antara lain :

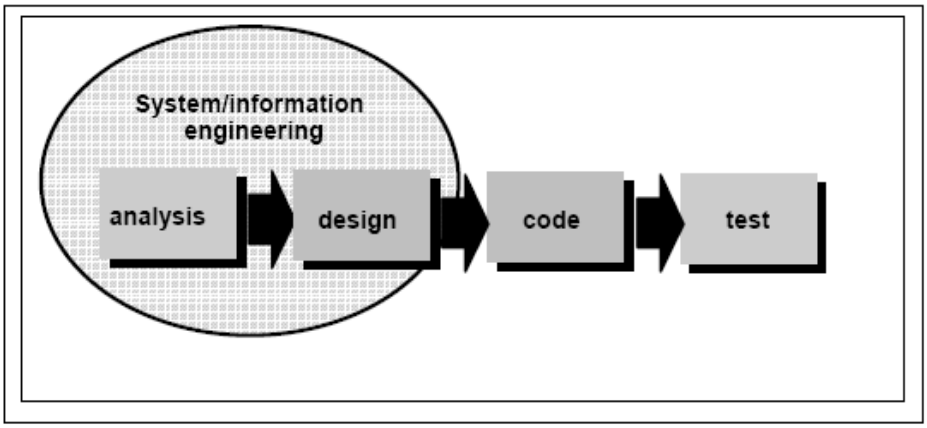
  - Wawancara (interview)
  - Observasi (observation)
  - Daftar Pertanyaan (questionaire)
  - Pengumpulan sampel (sampling)
  - Teknik analisis biaya / manfaat (cost effectiveness analysis atau cost benefit analysis)
  - Teknik menjalankan rapat
  - Teknik Inspeksi (Walkthrough)

WATERFALL

Model ini adalah model klasik yang bersifat sistematis, berurutan dalam membangun software. Berikut ini ada dua gambaran dari

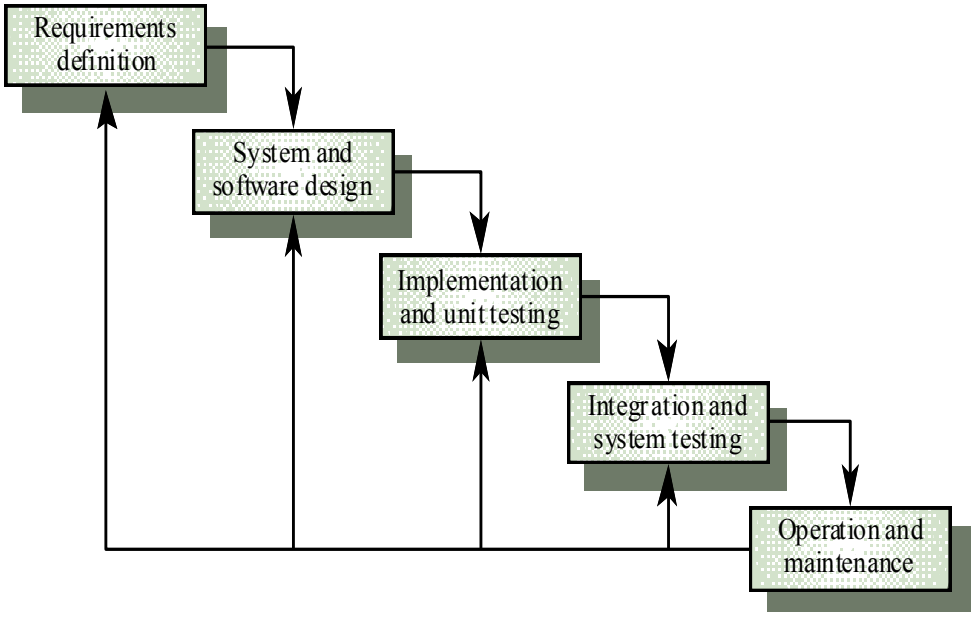
waterfall model. Sekalipun keduanya menggunakan nama-nama fase yang berbeda, namun sama dalam intinya.

Fase-fase dalam Waterfall Model menurut referensi Pressman:



Gambar 2.2 Fase-fase dalam Waterfall Model menurut referensi Pressma

Fase-fase dalam Waterfall Model menurut referensi Sommerville :



Gambar 2.3 Fase-fase dalam Waterfall Model menurut referensi Sommerville.

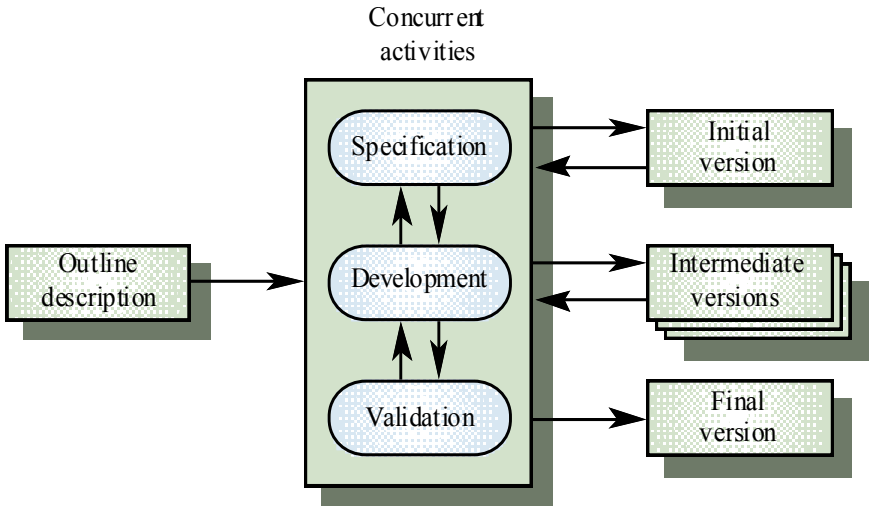
- **Requirements analysis and definition:** Mengumpulkan kebutuhan secara lengkap kemudian kemudian dianalisis dan didefinisikan kebutuhan yang harus dipenuhi oleh program yang akan dibangun. Fase ini harus dikerjakan secara lengkap untuk bisa menghasilkan desain yang lengkap.
- **System and software design:** Desain dikerjakan setelah kebutuhan selesai dikumpulkan secara lengkap.
- **Implementation and unit testing:** desain program diterjemahkan ke dalamnkode-kode dengan menggunakan bahasa pemrograman yang sudahnditentukan. Program yang dibangun langsung diuji baik secara unit.
- **Integration and system testing:** Penyatuan unit-unit program kemudiandiuji secara keseluruhan (system testing).
- **Operation and maintenance:** mengoperasikan program dilingkungannya dan melakukan pemeliharaan, seperti penyesuaian atau perubahan karena adaptasi dengan situasi sebenarnya.

a. Evolutionary Software Process Models / Pengembangan Evolusioner

Berdasarkan pada ide untuk mengembangkan implementasi awal, memperlihatkannya kepada *user* untuk dikomentari, dan memperbaikinya versi demi versi sampai sistem yang memenuhi

persyaratan diperoleh. Tidak ada kegiatan spesifikasi, pengembangan, dan validasi yang terpisah. Kegiatan-kegiatan ini dilakukan pada saat yang bersamaan dengan umpan balik yang cepat untuk masing-masing kegiatan.

Gambar Model Pengembangan Evolusioner



Gambar 2.4 Model Pengembangan Evolusioner

Dua model dalam evolutionary software process model adalah:

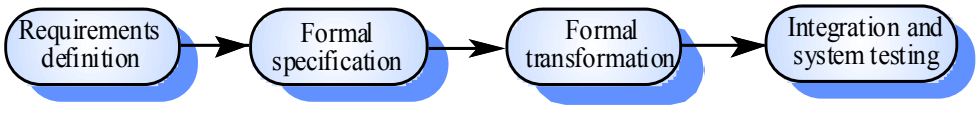
1. Pengembangan Eksplotari → Tujuan proses ini adalah bekerja dengan pelanggan untuk menyelidiki persyaratan mereka dan mengirimkan sistem akhir. Harusnya diawali dengan kebutuhan yang sudah dimengerti.

2. Prototipe yang dapat dibuang *(throw-away)* → Berkonsentrasi pada eksperimen, dengan persyaratan pelanggan yang tidak dipahami dengan baik.

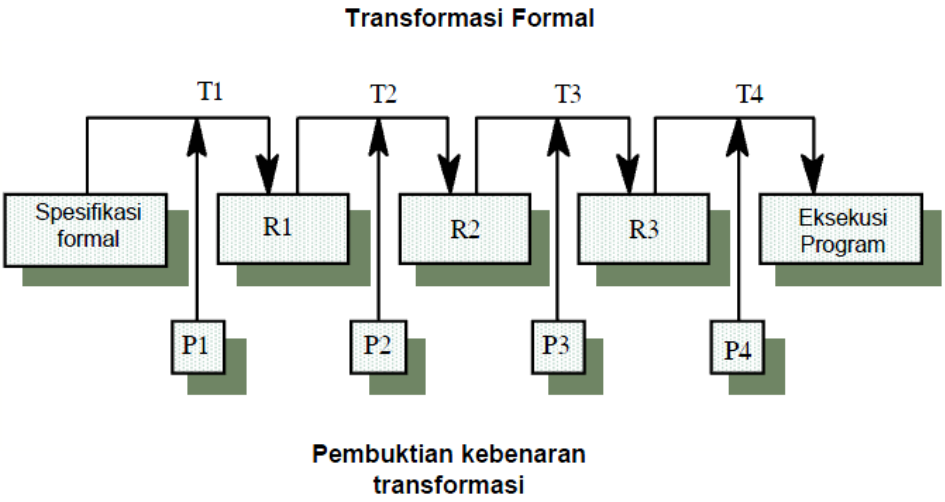
a. Model Pengembangan Sistem Formal

- Berbasis transformasi dari spesifikasi matematis melalui representasi yang berbeda untuk program yg dapat dieksekusi
- Transformasi adalah 'pemelihara kebenaran' sehingga dapat menunjukkan program sesuai spesifikasinya
- Merupakan pendekatan 'Cleanroom' untuk pengembangan software

Gambar pengembangan Sistem Formal :



Gambar 2.5 Pengembangan Sistem Formal

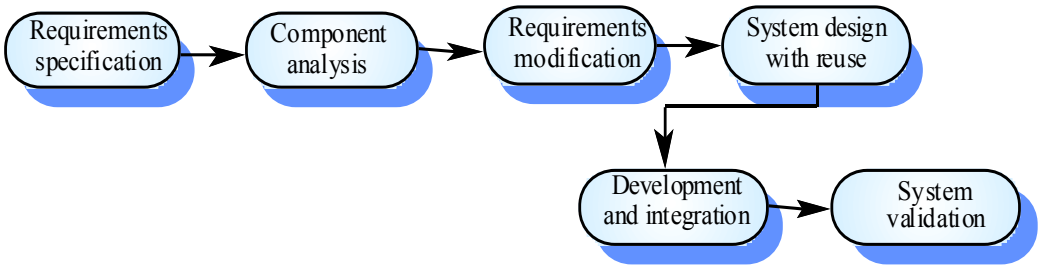


Gambar 2.6 Transformasi Formal

b. Model Pengembangan Berorientasi Pemakaian Ulang *(Re-Usable)*

- Bergantung pada sejumlah besar komponen perangkat lunak yang dapat dipakai ulang, yang bisa didapat, dan berapa kerangka kerja integrasi untuk komponen-komponen ini.
- Komponen-komponen ini dapat juga sistem yang disebut COTS *(Commercial Off-The-Shelf Systems*/Sistem Siap Beli Komersial) yang dapat digunakan untuk memberikan fungsionalitas khusus seperti format teks, perhitungan numerik,dll.

Gambar Model Pengembangan Berorientasi Pemakaian Ulang *(Re-Usable) :*



Gambar 2.7 Model Pengembangan Berorientasi Pemakaian Ulang

Fase-fase Re-Usable :

a. Analisis Komponen → Spesifikasi persyaratan telah diketahui, komponen2 untuk implementasi spesifikasi tersebut akan dicari. Biasanya, tidak ada kesesuaian yang tepat dan komponen yang dapat dipakai hanya memberikan sebagian dari fungsionaliyas yang dibutuhkan.
b. Modifikasi Persyaratan → Persyaratan dianalisis menggunakan informasi tentang komponen yang didapat, kemudian dimodifikasi untuk merefleksikan komponen yang ada. Jika modifikasi tidak mungkin dilakukan, maka kegiatan analisis komponen bisa diulang untuk mencari solusi alternatif.

c. Perancangan sistem dengan pemakaian ulang → Kerangka kerja sistem dirancang, atau kerangka kerja yang telah ada dipakai ulang.
d. Pengembangan dan Integrasi → Perangkat Lunak yang tidak dapat dibeli akan dikembangkan dan komponen dan sistem COTS diintegrasikan untuk membantu sistem.

Karakteristik analisis dan Pengembangan berorientasi objek

Metodologi pengembangan sistem berorientasi objek mempunyai tiga karakteristik utama :

a. Encapsulation (Pengkapsulan)

- Encapsulation merupakan dasar untuk pembatasan ruang lingkup program terhadap data yang diproses.
- Data dan prosedur atau fungsi dikemas bersama-sama dalam suatu objek, sehingga prosedur atau fungsi lain dari luar tidak dapat mengaksesnya.
- Data terlindung dari prosedur atau objek lain, kecuali prosedur yang berada dalam objek itu sendiri.

b. Inheritance (Pewarisan)

- Inheritance adalah teknik yang menyatakan bahwa anak dari objek akan mewarisi data/atribut dan metode dari induknya langsung. Atribut dan metode dari objek dari objek induk diturunkan kepada anak objek, demikian seterusnya.
- Inheritance mempunyai arti bahwa atribut dan operasi yang dimiliki bersama di anatara kelas yang mempunyai hubungan secara hirarki. Suatu kelas dapat ditentukan secara umum, kemudian ditentukan spesifik menjadi subkelas. Setiap subkelas mempunyai hubungan atau mewarisi semua sifat yang dimiliki oleh kelas induknya, dan ditambah dengan sifat unik yang dimilikinya.
- Kelas Objek dapat didefinisikan atribut dan service dari kelas Objek lainnya. Inheritance menggambarkan generalisasi sebuah kelas

  Contoh :

  - *Sedan* dan *Sepeda Motor* adalah *subkelas* dari *Kendaraan Bermotor*.

  - Kedua subkelas *mewarisi* sifat yang dimiliki oleh *Kendaraan Bermotor*, yaitu mempunyai mesin dan dapat berjalan.

- Kedua subkelas *mempunyai sifat masing-masing yang berbeda*, misalnya jumlah roda, dan kemampuan untuk berjalan mundur yang tidak dimiliki oleh sepeda motor.

c. Polymorphism (Polmorfisme)

- Polimorfisme yaitu konsep yang menyatakan bahwa seuatu yang sama dapat mempunyai bentuk dan perilaku berbeda.
- Polimorfisme mempunyai arti bahwa operasi yang sama mungkin mempunyai perbedaan dalam kelas yang berbeda.

2.4.3 Alat bantu Analisis

1. UNIFIED MODELING LANGUAGE (UML)

UML *(Unified Modeling Language)* adalah metode pemodelan secara visual sebagai sarana untuk merancang dan atau membuat software berorientasi objek. Karena UML ini merupakan bahasa visual untuk pemodelan bahasa berorientasi objek, maka semua elemen dan diagram berbasiskan pada paradigma *object*

oriented. UML adalah salah satu *tool* / model untuk merancang pengembangan software yang berbasis *object oriented*. UML sendiri juga memberikan standar penulisan sebuah sistem *blue print*, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema database, dan komponenkomponen yang diperlukan dalam sistem software. UML sebagai sebuah bahasa yang memberikan *vocabulary* dan tatanan penulisan kata-kata dalam 'MS Word' untuk kegunaan komunikasi. Sebuah bahasa model adalah sebuah bahasa yang mempunyai *vocabulary* dan konsep tatanan / aturan penulisan serta secara fisik mempresentasikan dari sebuah sistem. UML adalah sebuah bahasa standar untuk pengembangan sebuah software yang dapat menyampaikan bagaimana membuat dan membentuk model-model, tetapi tidak menyampaikan apa dan kapan model yang seharusnya dibuat yang merupakan salah satu proses implementasi pengembangan software. UML tidak hanya merupakan sebuah bahasa pemograman visual saja, namun juga dapat secara langsung dihubungkan ke berbagai bahasa pemograman, seperti JAVA, C++, Visual Basic, atau bahkan dihubungkan secara langsung ke dalam sebuah object-oriented database. Begitu juga mengenai pendokumentasian dapat dilakukan seperti; *requirements*, arsitektur, *design*, *source code*, *project plan*, *tests*, dan *prototypes*. Untuk dapat memahami UML membutuhkan bentuk konsep dari

sebuah bahasa model, dan mempelajari 3 (tiga) elemen utama dari UML, seperti *building block*, aturan-aturan yang menyatakan bagaimana *building block* diletakkan secara bersamaan, dan beberapa mekanisme umum (*common*). **Building blocks**

Tiga macam yang terdapat dalam building block adalah :

• ***Benda/Things*** Adalah abstraksi yang pertama dalam sebuah model

• ***Hubungan/Relationships*** Sebagai alat komunikasi dari benda-benda

• ***Bagan/Diagrams*** Sebagai kumpulan / group dari benda-benda/things

Benda/Things Adalah hal yang sangat mendasar dalam model UML, juga merupakan bagian paling statik dari sebuah model, serta menjelaskan elemenelemen lainnya dari sebuah konsep dan atau fisik. Bentuk dari beberapa benda / thing adalah sebagai berikut

• ***Classes,*** yang diuraikan sebagai sekelompok dari *object* yang mempunyai *atribute*, operasi, hubungan yang semantik. Sebuah kelas mengimplementasikan 1 atau lebih *interfaces*. Sebuah kelas

dapat digambarkan sebagai sebuah persegi panjang, yang mempunyai sebuah nama, *atribute*, dan metoda pengoperasiannya.

• ***Interfaces,*** merupakan sebuah antar-muka yang menghubungkan dan melayani antar kelas dan atau elemen. *Interface* / antar-muka mendefinisikan sebuah set / kelompok dari spesifikasi pengoperasian, umumnya digambarkan dengan sebuah lingkaran yang disertai dengan namanya. Sebuah antar-muka berdiri sendiri dan umumnya merupakan pelengkap dari kelas atau komponen.

• ***Collaboration,*** yang didefinisikan dengan interaksi dan sebuah kumpulan / kelompok dari kelas-kelas / elemen-elemen yang bekerja secara bersama-sama. *Collaborations* mempunyai struktur dan dimensi. Pemberian sebuah kelas memungkinkan berpartisipasi didalam beberapa *collaborations* dan digambarkan dengan sebuah '*elips*' dengan garis terpotong-potong.

• ***Use cases,*** adalah rangkaian/uraian sekelompok yang saling terkait dan membentuk sistem secara teratur yang dilakukan atau diawasi oleh sebuah aktor. '*use case*' digunakan untuk membentuk tingkah-laku benda / *things* dalam sebuah model serta di realisasikan oleh sebuah *collaboration*. Umumnya '*use case*' digambarkan dengan sebuah '*elips*' dengan garis yang solid, biasanya mengandung nama.

• ***Nodes,*** merupakan fisik dari elemen-elemen yang ada pada saat dijalankannya sebuah sistem, contohnya adalaha sebuah komputer, umumnya mempunyai sedikitnya *memory* dan *processor*. Sekelompok komponen mungkin terletak pada sebuah node dan juga mungkin akan berpindah dari node satu ke node lainnya. Umumnya node ini digambarkan seperti kubus serta hanya mengandung namanya.

Hubungan / Relationship

Ada 4 macam hubungan didalam penggunaan UML, yaitu;

• ***Dependency,*** adalah hubungan semantik antara dua benda/*things* yang mana sebuah benda berubah mengakibatkan benda satunya akan berubah pula. Umumnya sebuah *dependency* digambarkan sebuah panah dengan garis terputusputus.

• **Association,** hubungan antar benda struktural yang terhubung diantara obyek. Kesatuan obyek yang terhubung merupakan hubungan khusus, yang menggambarkan sebuah hubungan struktural diantara seluruh atau sebagian. Umumnya *assosiation*

digambarkan dengan sebuah garis yang dilengkapi dengan sebuah label, nama, dan status hubungannya.

• **Generalizations,** adalah menggambarkan hubungan khusus dalam obyek anak/*child* yang menggantikan obyek *parent* / induk. Dalam hal ini, obyek anak memberikan pengaruhnya dalam hal struktur dan tingkah lakunya kepada obyek induk. Digambarkan dengan garis panah.

• **Realizations,** merupakan hubungan semantik antara pengelompokkan yang menjamin adanya ikatan diantaranya. Hubungan ini dapat diwujudkan diantara *interface* dan kelas atau *elements*, serta antara *use cases* dan *collaborations*. Model dari sebuah hubungan *realization*.

1. **Diagram - Diagram Yang Terdapat Pada UML**

UML sendiri terdiri atas pengelompokkan diagram-diagram sistem menurut aspek atau sudut pandang tertentu. Diagram adalah yang menggambarkan permasalahan maupun solusi dari permasalahan suatu model.

UML mempunyai 9 diagram, yaitu;

• Diagram Use Case

• Diagram Class

• Diagram Package

• Diagram Sequence

• Diagram Collaboration

• Diagram StateChart

• Diagram Activity

• Diagram Deployment

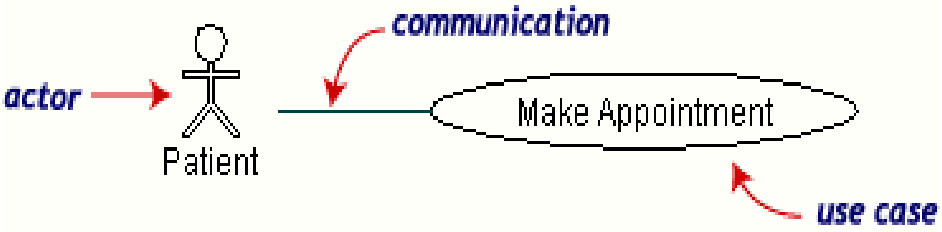
Semakin kompleks bentukan sistem yang akan dibuat, maka semakin sulit komunikasi antara orang-orang yang saling terkait dalam pembuatan dan pengembangan software yang akan dibuat. Pada masa lalu, UML mempunyai peranan sebagai software blueprint (gambaran) language untuk analisis sistem, designer, dan programmer. Sedangkan pada saat ini, merupakan bagian dari software trade (bisnis software). UML memberikan jalur komunikasi dari sistem analis kemudian designer, lalu programmer mengenai rancangan software yang akan dikerjakan. Salah satu pemecahan masalah *Object Oriented* adalah dengan menggunakan UML. Oleh karena itu orang-orang yang berminat dalam mempelajari UML harus mengetahui dasar-dasar mengenai *Object*

Oriented Solving (pemecahan masalah OO). Tahap pertama, pembentukan model. ***Model*** adalah gambaran abstrak dari suatu dasar masalah. Dan dunia nyata atau tempat dimana masalah itu timbul bisa disebut dengan ***domain.*** Model mengandung obyek-obyek yang beraktifitas dengan saling mengirimkan messages (pesan-pesan). Obyek mempunyai sesuatu yang diketahui (atribut /attributes) dan sesuatu yang dil akukan (behaviors atau operations). Attributes hanya berlaku dalam ruang lingkup obyek itu sendiri (state). Lalu "blue print" dari suatu obyek adalah Classes (kelas). Obyek merupakan bagian-bagian dari kelas.

Diagram Use Case

Diagram Use Case menggambarkan apa saja aktifitas yang dilakukan oleh suatu sistem dari sudut pandang pengamatan luar. yang menjadi persoalan itu *apa yang dilakukan* bukan *bagaimana melakukannya*. Diagram Use Case dekat kaitannya dengan kejadian-kejadian. Kejadian (scenario) merupakan contoh apa yang terjadi ketika seseorang berinteraksi dengan sistem. untuk lebih memperjelas lihat gambaran suatu peristiwa untuk sebuah klinik kesehatan di bawah ini : "Pasien menghubungi klinik untuk membuat janji (*appointment*) dalam pemeriksaan tahunan.

Receptionist mendapatkan waktu yang luang pada buku jadwal dan memasukkan janji tersebut ke dalam waktu luang itu."



Gambar 2.8. Contoh Diagram Use Case

Diagram Use Case berguna dalam tiga hal :

• Menjelaskan fasilitas yang ada (*requirements*) Use Case baru selalu menghasilkan

fasilitas baru ketika sistem di analisa, dan *design* menjadi lebih jelas.

• Komunikas dengan klien Penggunaan notasi dan simbol dalam diagram Use Case membuat pengembang lebih mudah berkomunikasi dengan klienkliennya.

• Membuat test dari kasus-kasus secara umum Kumpulan dari kejadian-kejadian untuk Use Case bisa dilakukan test kasus layak untuk kejadian-kejadian tersebut.

Diagram Class

Diagram Class memberikan pandangan secara luas dari suatu sistem dengan menunjukan kelas-kelasnya dan hubungan mereka. Diagram Class bersifat statis; *menggambarkan hubungan apa yang terjadi bukan apa yang terjadi jika mereka berhubungan*. Diagram Class mempunyai 3 macam relationalships (hubungan), sebagai berikut :

• **Association**

Suatu hubungan antara bagian dari dua kelas. Terjadi *association* antara dua kelas jika salah satu bagian dari kelas mengetahui yang lainnya dalam melakukan suatu kegiatan. Di dalam diagram, sebuah *association* adalah penghubung yang menghubungkan dua kelas.

• **Aggregation**

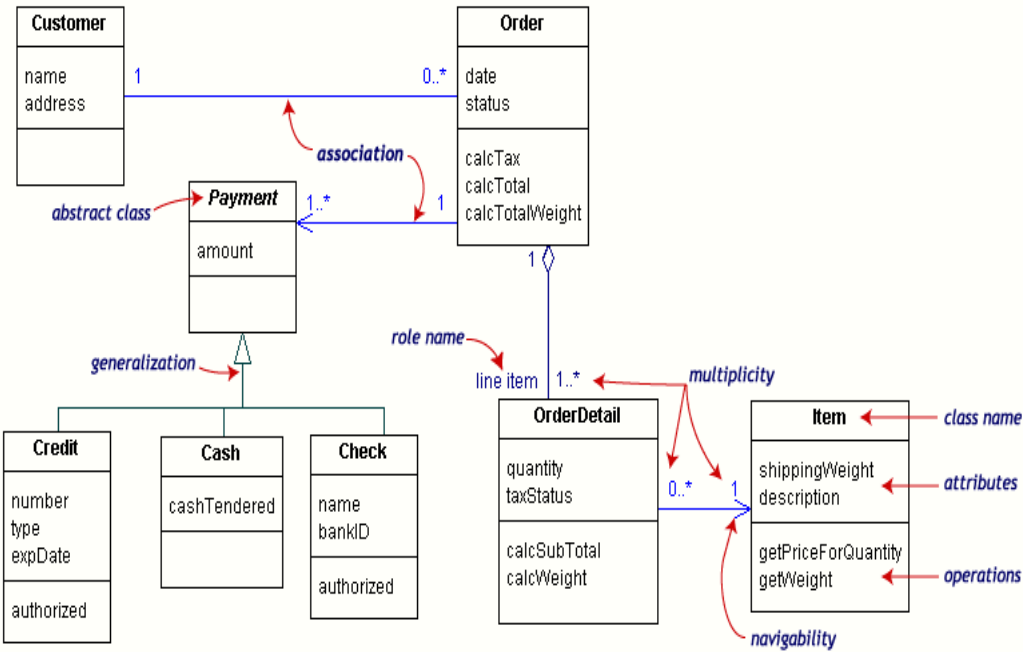
Suatu *association* dimana salah satu kelasnya merupakan bagian dari suatu kumpulan. *Aggregation* memiliki titik pusat yang

mencakup keseluruhan bagian. Sebagai contoh : OrderDetail merupakan kumpulan dari Order.

• **Generalization**

Suatu hubungan turunan dengan mengasumsikan satu kelas merupakan suatu *superClass* (kelas super) dari kelas yang lain. *Generalization* memiliki tingkatan yang berpusat pada *superClass*. *Contoh :* Payment adalah *superClass* dari Cash, Check, dan Credit. Untuk tambahan bahwa *association* mempunyai 2 titik. Salah satu titik bisa memiliki label untuk menjelaskan *association* tersebut. *Contoh :* OrderDetail adalah line Item untuk setiap permintaan. Panah *navigability* (pengatur alur arah) dalam suatu *association* menggambarkan arah mana *association* dapat ditransfer atau disusun. Seperti dalam contoh : OrderDetail dapat disusun dari item-nya, namun tidak bisa sebaliknya. Panah ini juga menjelaskan siapa "memiliki" implementasi dari *association*; dalam kasus ini OrderDetail memiliki Item. *Association* tanpa arah panah merupakan *bidirectional* (bolak-balik). *Multiplicity* dari suatu titik *association* adalah angka kemungkinan bagian dari hubungan kelas dengan single *instance* (bagian) pada titik yang lain. *Multiplicity* berupa single number (angka tunggal) atau range number (angka batasan). Pada contoh, hanya bisa satu 'Customer' untuk setiap

‘Order’, tapi satu ‘Customer’ hanya bisa memiliki beberapa ‘Order’. Tabel di bawah mengenai *multiplicity* yang sering digunakan: Tabel *Multiplicity* **Multiplicities Artinya 0..1** Nol atau satu bagian. Notasi ***n . . m*** menerangkan ***n*** sampai ***m*** bagian. **0..*** *or* ***** Tak hingga pada jangkauan bagian (termasuk kosong). **1** Tepat satu bagian **1..*** Sedikitnya hanya satu bagian Setiap diagram Class memiliki *Class* (kelas), *association*, dan *multiplicity*. Sedangkan *navigability* (alur arah) dan *role* (kegiatan) merupakan **optional** (tidak diharuskan).



Gambar 2 .9 Contoh Diagram Class transaksi Pembelian barang

Package dan Object

Untuk mengatur pengorganisasian diagram Class yang *kompleks*, dapat dilakukan pengelompokan kelas-kelas berupa *packag*e (paketpaket). *Package* adalah kumpulan elemen-elemen logika UML. Gambar di bawah ini mengenai model bisnis dengan pengelompokan kelas-kelas dalam bentuk paket-paket :

Ada jenis khusus dari diagram Class yaitu diagram Object. Kegunaannya untuk penjelasan yang sedikit dengan relasi yang sulit, khususnya relasi rekursif. Lihat gambar dibawah, diagram Class kecil menunjukkan bahwa ‘department’ dapat mengandung banyak ‘department’ yang lain. Setiap tingkatan pada diagram berpengaruh pada *single instance* (bagian tunggal). Nama bagian digarisbawahi dalam diagram UML. Untuk *Class name* (nama kelas) maupun *instance name* (nama bagian) bisa mengambil dari diagram Object selama arti diagram tersebut masih jelas.

Diagram Sequence

Diagram Class dan diagram Object merupakan suatu gambaran *model statis*. Namun ada juga yang bersifat *dinamis*, seperti **Diagram Interaction**. Diagram sequence merupakan salah satu diagram Interaction yang menjelaskan bagaimana suatu operasi itu

dilakukan; *message*(pesan) apa yang dikirim dan kapan pelaksanaannya. Diagram ini diatur berdasarkan waktu. Obyek-obyek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu.

Di bawah ini adalah diagram Sequence untuk pembuatan Hotel Reservation. Obyek yang mengawali urutan *message* adalah ‘aReservation Window’. Reservation window’ mengirim pesan makeReservation ke ‘HotelChain’. Kemudian ‘HotelChain’ mengirim pesan yang sama ke ‘Hotel’. Bila ‘Hotel’ punya kamar kosong, maka dibuat ‘Reservation’ dan ‘Confirmation’. *Lifeline* adalah garis dot (putus-putus) vertikal pada gambar, menerangkan waktu terjadinya suatu obyek. Setiap panah yang ada adalah pemanggilan suatu pesan. Panah berasal dari pengirim ke bagian paling atas dari batang kegiatan (*activation bar*) dari suatu pesan pada *lifeline* penerima. *Activation bar* menerangkan lamanya suatu pesan diproses.

Pada gambar diagram , terlihat bahwa ‘Hotel’ telah melakukan pemanggilan diri sendiri untuk pemeriksaan jika ada kamar kosong. Bila benar, maka ‘Hotel’ membuat ‘Reservation’ dan ‘Confirmation’. Pemanggilan diri sendiri disebut dengan *iterasi*. *Expression* yeng dikurung dengan “[ ]”, adalah *condition* (keadaan

kondisi). Pada diagram dapat dibuat *note* (catatan). Pada gambar, terlihat seperti selembar kertas yang berisikan teks. *Note* bisa diletakan dimana saja pada diagram UML.

Diagram Collaboration

Diagram Collaboration juga merupakan *diagram interaction*. Diagram membawa informasi yang sama dengan diagram Sequence, tetapi lebih memusatkan atau memfokuskan pada kegiatan obyek dari waktu pesan itu dikirimkan Kotak kegiatan obyek diberi label dengan nama kelas atau obyek (atau keduanya). Nama kelas dibatasi dengan *colons /titik dua* (:). Setiap pesan pada diagram Collaboration mempunyai angka yang terurut. Pesan yang tingkatannya tertinggi adalah angka 1. Pesan yang berada pada tingkat yang sama memiliki *prefix* yang sama, namun *suffix* berbeda bergantung pada posisinya; hanya untuk angka 1, 2, dan seterusnya.

Diagram StateChart

Behaviors dan *state* dimiliki oleh obyek. Keadaan dari suatu obyek bergantung pada kegiatan dan keadaan yang berlaku pada saat itu. Diagram StateChart menunjukan kemungkinan dari keadaan obyek dan proses yang menyebabkan perubahan pada keadaannya. Untuk lebih jelas, contoh yang digunakan model diagram untuk login yang merupakan bagian dari Online Banking System. Logging in terdiri atas masukan input Social Security Number dan Personal Id Number yang berlaku, lalu memutuskan kesahan dari informasi tersebut.

Logging in dapat dibagi menjadi empat tahapan proses, yaitu :

• Getting SSN (masukkan SSN)

• Getting PIN (masukkan PIN)

• Validating (periksa kesahannya)

• Rejecting (keluar)

Proses peralihan digambarkan dengan panah dari satu state ke yang lainnya. *Event* (peristiwa) atau *condition* (keadaan) yang menyebabkan perubahan dituliskan pada samping panah. Diagram ini mengandung dua *self-transition* (transisi sendiri), satu pada getting SSN dan lainnya pada getting PIN. Keadaan awal *Start* (*black circle /*lingkar hitam) adalah *dummy* (model) untuk memulai

action (kegiatan). Keadaan akhir juga keadaan model yang menghentikan kegiatan. Aksi yang terjadi sebagai hasil dari suatu peristiwa atau keadaan ditandai dalam bentuk /action. Pada Validating State, obyek tidak menunggu peristiwa dari luar untuk menyebabkan suatu perubahan. Sebagai gantinya melakukan suatu *activity* (aktifitas). Hasil dari aktifitas tersebut menentukan keadaan berikutnya dari obyek tersebut.

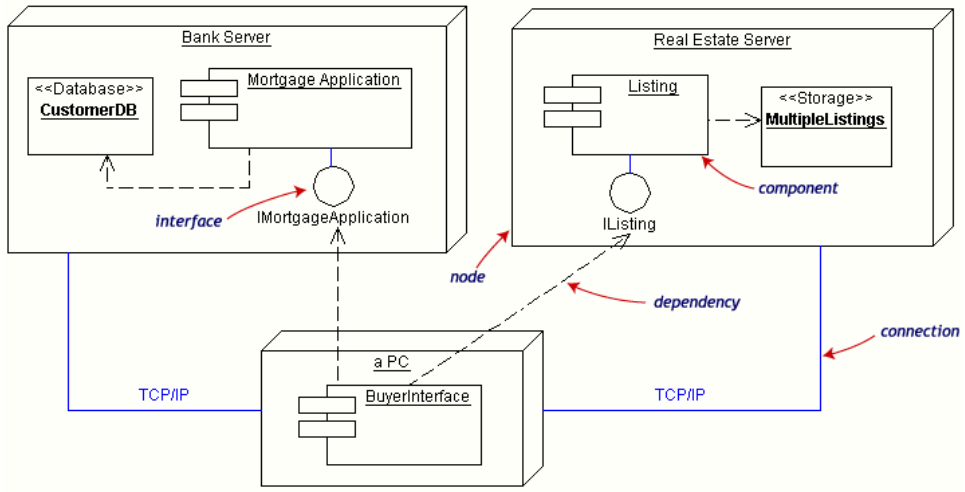
Diagram Activity

Pada dasarnya diagram Activity sering digunakan oleh *flowchart*. Diagram ini berhubungan dengan diagram Statechart. Diagram Statechart berfokus pada *obyek yang dalam suatu proses* (atau proses menjadi suatu obyek), diagram Activity berfokus pada *aktifitas-aktifitas yang terjadi yang terkait dalam suatu proses tunggal*. Jadi dengan kata lain, diagram ini menunjukkan bagaimana aktifitas-aktifitas tersebut bergantung satu sama lain. Sebagai contoh, perhatikan proses yang terjadi. “Pengambilan uang dari bank melalui ATM.” Ada tiga aktifitas kelas (orang, dan lainnya) yang terkait, yaitu : Customer, ATM, and Bank. Proses berawal dari lingkaran start hitam pada bagian atas dan berakhir di pusat lingkaran stop hitam/putih pada bagian bawah. Aktivitas digambarkan dalam bentuk kotak persegi. Contoh Diagram

Activity ‘Pengambilan Uang melalui ATM’ Diagram Activity dapat dibagi menjadi beberapa jalur kelompok yang menunjukkan obyek yang mana yang bertanggung jawab untuk suatu aktifitas. Peralihan tunggal (*single transition*) timbul dari setiap adanya *activity* (aktifitas), yang saling menghubungi pada aktifitas berikutnya. Sebuah *transition* (transisi) dapat membuat cabang ke dua atau lebih percabangan *exclusive transition* (transisi eksklusif). Label *Guard Expression* (ada di dalam [ ]) yang menerangkan output (keluaran) dari percabangan. percabangan akan menghasilkan bentuk menyerupai bentuk intan. *transition* bisa bercabang menjadi beberapa aktifitas paralel yang disebut **Fork.** *Fork* beserta *join* (gabungan dari hasil output *fork*) dalam diagram berbentuk *solid bar* (batang penuh).

Diagram Component dan Deployment

Component adalah sebuah *code module* (kode-kode module). Diagram Component merupakan *fisik sebenarnya dari diagram Class*. Diagram Deployment menerangkan bahwa *konfigurasi fisik software* dan *hardware*. Gambar 3 menerangkan hubungan sekitar komponen software dan hardware yang berperan dalam ruang lingkup real estate.



Gambar 2.10 Contoh Diagram Deployment 'Sistem Real Estate'

Fisik hardware berbentuk seperti *node-node*. Setiap komponen merupakan bagian dari *node*. Pada gambar komponen berbentuk dua kotak tersusun yang terletak di sebelah kiri atas.

2. Perancangan Basis Data

Basis data adalah satu kelompok organisasi data yang terpusat. Basis data secara umum dianalogikan sebagai lemari dokumen atau sekumpulan lemari dokumen (Beynon-Davies, 1991) Basis data diorganisasikan sebagai tempat penyimpanan data.

Entity Relationship Diagram (ERD)

Diagram E-R digunakan untuk mengembangkan model tingkat tinggi system, yang menggambarkan sebagian besar obyek system serta interaksi antara obyek dan atribut-atributnya
(Hawryszieycs, 1990).

Kardinalitas

Meskipun diagram E-R sudah menggambarkan informasi tentang system, namun masih ada atribut tambahan yang dapat ditampilkan dengan diagram E-R untuk melengkapi pemodelan, atribut tersebut adalah kardinalitas relasi. Kardinalitas menunjukan nomor relasi yang dimiliki oleh suatu entitas (Hawryszieycs, 1990).

Normalisasi

Normalisasi merupakan proses dekomposisi table agar terbentuk table normal. Bentuk-bentuk normalisasi yang digunakan adalah:

1. Bentuk Normal Kesatu (1 NF / First Normal Form)

Syarat agar suatu relasi dapat dikatakan memenuhi bentuk normal kesatu antara lain jika tidak mengandung repeat group dan harus atomic.

2. Bentuk Normal Kedua (2 NF / Second Normal Form)

Suatu relasi akan memenuhi bentuk normal kedua apabila telah memenuhi bentuk normal kesatu dan semua atribut bukan kunci harus secara fungsi bergantung penuh pada atribut kunci.

3. Bentuk Normal Ketiga (3 NF / Third Normal Form)

Suatu relasi dapat memenuhi bentuk normal ketiga jika telah memenuhi bentuk normal kedua dan tidak mengandung "Transitive Functional Dependency" (ketergantungan transitif/tidak langsung).