# BAB II

# TINJAUAN TEORI

Teori merupakan dasar yang digunakan dalam penyusunan sistem informasi yang dibangun. Pada bab ini akan dijelaskan beberapa teori yang berkaitan dengan permasalahan yang akan dibahas sebagai dasar dalam pemahaman dalam sebuah sistem serta metoda yang dipakai untuk kegiatan pengembangan terhadap sistem itu sendiri.

## 2.1 Konsep Dasar Sistem

Pendekatan sistem yang merupakan jaringan kerja dari prosedur lebih menekankan urutan-urutan operasi di dalam sistem. Pendekatan yang menekankan pada prosedur mendefinisikan sistem sebagai berikut :

Pengertian Menurut JOG[6] :

*"Sistem adalah suatu jaringan kerja dari prosedur- prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran yang tertentu."*

Jaringan kerjanya berupa subsistem yaitu bagian kecil dari sistem maupun prosedur yang merupakan cara kerja dasi sistem itu sendiri.

Pendekatan yang menekankan pada elemen atau komponennya mendefinisikan sistem sebagai berikut :

Menurut JOG[6] :

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

Adapun tujuan dari sistem tersebut adalah untuk membantu manajemen dalam pengambilan keputusan. Dimana elemen-elemen tersebut dapat berupa subsistem atau bagian sistem itu sendiri.

Terdapat juga pengertian sistem yang lain yaitu :

Menurut ZUL [11 ] :

*“Sistem adalah himpunan sesuatu “benda” nyata atau abstrak yang terdiri dari bagian-bagian atau komponen-komponen yang saling berkaitan, berhubungan, berketergantungan, dan saling mendukung, yang secara keseluruhan bersatu dalam satu kesatuan untuk mencapai tujuan tertentu secara efisien dan efektif.”*

Adapun tujuan dari sistem tersebut adalah untuk membantu manajemen dalam pengambilan keputusan secara efektif dan efisien. Dimana bagian-bagian atau komponen-komponen tersebut dapat berupa subsistem atau bagian sistem itu sendiri.

Dari definisi diatas dapat pula disimpulkan bahwa sistem adalah : Kumpulan dari elemen-elemen yang saling berhubungan, berkaitan, berhubungan, berketergantungan dan saling mendukung membentuk satu kesatuan atau organisasi untuk mencapai tujuan dan sasaran tertentu secara efektif dan efisien.

### 2.1.1 Karakteristik sistem

Karakteristik sistem menurut Jogiyanto H.M dalam bukunya yang berjudul *Pengenalan Komputer* adalah:

Menurut JOG [6] :

*Sistem mempunyai karakteristik atau sifat-sifat yang tertentu, yaitu mempunyai komponen (components), batas sistem (boundary), lingkungan luar sistem (environment), penghubung (interface), masukan (input), keluaran (output), pengolah (proses), sasaran (objectives),atau tujuan (goal).*

Berikut ini penjelasan dari karakteristik sistem diatas:

a. Komponen Sistem

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, artinya saling bekerjasama membentuk satu kesatuan. Komponen sistem dapat berupa subsistem atau bagian dari sistem. Setiap subsistem mempunyai sifat-sifat dari sistem yang menjalankan suatu fungsi tertentu dan mempengaruhi proses secara keseluruhan. Suatu sistem dapat mempunyai suatu sistem yang lebih besar yang disebut dengan supra sistem.

b. Batas Sistem

   Merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainya atau dengan lingkungan luarnya.

c. Lingkungan Luar Sistem

   Adalah apapun diluar batas dari sistem yang mempengaruhi operasi sistem.

d. Penghubung *(Interface)* Sistem

Merupakan media penghubung antara satu subsistem dengan subsistem lainya.

e. Masukan *(Input)* Sistem

Adalah energi yang dimasukkan kedalam sistem.

f. Keluaran *(Output*) Sistem

Adalah hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan.

g. Pengolah Sistem (Proses)

Suatu sistem dapat mempunyai suatu bagian pengolah yang akan mengubah masukan menjadi keluaran.

h. Sasaran (*Objective*s) Sistem

Suatu sistem pasti mempunyai tujuan atau sasaran, bila tidak memiliki sasaran maka operasi sistem tidak ada gunanya. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuan.

### 2.1.2 Klasifikasi Sistem

Sistem juga dapat diklasifikasikan dari beberapa sudut pandang, klasifikasi sistem menurut Jogiyanto H. M (1988 : 779) diantaranya adalah sebagai berikut :

1. Sistem Abstrak dan Sistem Fisik

Sistem abstrak adalah sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara phisik. Sistem fisik merupakan sistem yang ada secara fisik.

2. Sistem Alamiah dan Sistem Buatan Manusia

   Sistem alamiah adalah sistem yang terjadi melalui proses alam, tidak dibuat manusia. Sistem buatan manusia adalah sistem yang dirancang oleh manusia, sistem ini melibatkan interaksi antara manusia dengan mesin.

3. Sistem Tertentu dan Sistem Tak Tentu

   Sistem tertentu beroperasi dengan tingkah laku yang sudah dapat diprediksi. Interaksi antara bagian-bagiannya dapat dideteksi dengan pasti. Sistem tak tentu adalah sistem yang kondisi masa depannya tidak dapat diprediksi karena mengandung unsur probabilitas.

4. Sistem Tertutup dan Sistem Terbuka

   Sistem tertutup adalah sistem yang tidak berhubungan dan tidak terpengaruh dengan lingkungan luarnya. Sistem terbuka adalah sistem yang berhubungan dan terpengaruh dengn lingkungan luarnya.

## 2.2 Konsep Dasar Informasi

Informasi mempunyai peranan yang sangat penting dalam organisasi. Suatu organisasi yang kurang mendapatkan informasi akan luruh,kerdil dan pada akhirnya akan berakhir. Informasi dapat didefinisikan sebagai berikut :

Menurut JOG [6] :

"*Informasi adalah data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya*".

Data tersebut sebelumnya belum mempunyai makna apa-apa, namun setelah diolah maka sistem tersebut akan menjadi lebih berguna dan lebih berarti bagi orang yang membutuhkan informasi.

Menurut ZUL [ 11] :

*“ Informasi adalah data yang sudah diolah, atau dimanipulasi sesuai dengan keperluan tertentu.”*

Data tersebut sebelumnya belum mempunyai makna apa-apa, namun setelah diolah maka sistem tersebut akan menjadi lebih berguna dan lebih berarti untuk keperluan tertentu.

Dari definisi-definisi informasi diatas maka dapat diambil kesimpulan bahwa sumber informasi adalah data yang telah diolah yang menggambarkan suatu kejadian-kejadian dan kesatuan yang nyata.

### 2.2.1 Kualitas informasi

Kualitas informasi terdiri dari tiga hal, yaitu:

1. Akurat *(accurate)*

   Akurat berarti informasi harus bebas dari kesalahan-kesalahan dan tidak bisa atau menyesatkan, harus jelas mencerminkan maksudnya.

2. Tepat pada waktunya *(timeliness)*

   Tepat pada waktunya berarti informasi yang datang pada penerima tidak boleh terlambat.

3. Relevan *(relevance)*

   Relevan berarti informasi tersebut mempunyai manfaat untuk pemakainya.

### 2.2.2 Nilai informasi

Nilai dari informasi *(value of information)* ditentukan dari dua hal, yaitu manfaat dan biaya mendapatkanya. Suatu informasi dikatakan bernilai jika

manfaatnya lebih efektif dibandingkan biaya mendapatkannya. Lebih lanjut sebagian besar informasi tidak dapat persis ditaksir keuntungannya dengan satuan nilai uang, tetapi dapat ditaksir nilai efektivitasnya.

## 2.3 Konsep Dasar Sistem Informasi

Semakin majunya teknologi sekarang ini membuat setiap perusahaan atau instansi menerapkan suatu sistem informasi dengan mengikuti perkembangan zaman. Dengan diterapkan atau digunakannya sistem informasi yang baik maka akan mempermudah perusahaan atau instansi dalam mengoreksi kesalahan bahkan dapat dapat mengatasi kendala-kendala yang terjadi di dalam perusahan.

Informasi merupakan hal yang sangat penting bagi manajemen dalam mengambil keputusan. Informasi dapat diperoleh dari sistem informasi *(information system)* atau disebut juga dengan *processing system* atau *information processing system.* Sistem informasi dapat didefinisikan sebagi berikut :

Menurut JOG [6] :

*"Sistem informasi adalah suatu sistem didalam 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."*

Laporan – laporan yang diperlukan tersebut berupa hasil dari pengolahan taransaksi yang dilakukan.

Menurut Hall [1 ] :

*“ Sistem informasi adalah sebuah rangkaian prosedur formal dimana data dikelompokkan, di proses menjadi informasi , dan didistribusikan kepada pemakai.”*

Data di manipulasi untuk menghasilkan informasi kemudian disebarkan kepada pemakai yang membutuhkan informasi.

### 2.3.1 Komponen sistem informasi

Sistem informasi dapat terdiri dari komponen-komponen yang disebut dengan istilah blok bangunan *(building block)*, yaitu blok masukan *(input block)*, blok model *(model block)*, blok keluaran *(output block)*, blok teknologi *(technology block)*, blok dasis data *(database block*) dan blok kendali *(control block).*

Sebagai suatu sistem, keenam blok tersebut masing-masing saling berinteraksi satu dengan yang lainnya membentuk satu kesatuan untuk mencapai sasarannya:

**Blok Masukan / Blok Input**

Input mewakili data yang masuk kedalam sistem informasi, input berupa metode dan media untuk menangkap data, berupa dokumen-dokumen dasar.

**Blok Model / Blok Pengolahan**

Terdiri dari kombinasi prosedure, logika dan model matematik yang akan memanipulasi data input dan data yang tersimpan di basis data dengan cara yang sudah tertentu untuk menghasilkan keluaran yang diinginkan.

**Blok Output / Blok Keluaran**

Menghasilkan keluaran berupa informasi yang berkualitas dan dokumentasi yang berguna untuk semua tingkatan manajemen serta pemakai sistem lainnya.

**Blok Teknologi**

Teknologi merupakan "kotak alat" *(tool box)* yang digunakan untuk menerima input, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan keluar dan membantu pengendalian sistem secara keseluruhan.

Teknologi terdiri dari tiga bagian utama, yaitu teknisi *(brainware*), perangkat lunak *(software)* dan perangkat keras *(hardware).*

Teknisi dapat berupa orang-orang yang mengetahui teknologi dan mengoperasikannya. Contoh teknisi adalah operator komputer, pemrogram, operator pengolah kata, spesialis telekomunikasi, analis sistem, penyimpan data dan lain-lain.

**Blok Basis Data / Data Base Blok**

Basis data merupakan kumpulan dari data yang saling berhubungan satu sama lain, tersimpan di perangkat keras komputer dan dimanipulasi oleh perangkat lunak. Data perlu disimpan dalam basis data untuk penyediaan informasi lebih lanjut. Data di dalam basis data perlu diorganisasikan sedemikian rupa, agar informasi yang dihasilkan berkualitas. Organisasi dasar data yang baik juga berguna untuk efesiensi kapasitas penyimpannya. Basis data diakses atau dimanipulasi dengan menggunakan perangkat lunak paket yang disebut dengan DBMS *(Database Management systems).*

**Blok Kendali**

Merupakan penyaringan atau pencegahan untuk menghindarkan kesalahan input, proses, output dalam sistem. Agar sistem informasi dapat berjalan sesuai dengan yang diinginkan, maka perlu diterapkan pengendalian-pengendalian di dalamnya. Banyak hal yang dapat merusak sistem informasi, seperti misalnya bencana alam, api, temperatur, air, debu, kecurangan-kecurangan, kegagalan-kegagalan sistem itu sendiri, kesalahan-kesalahan, ketidakefisienan dan lain sebagainya. Beberapa pengendalian perlu dirancang dan diterapkan untuk meyakinkan bahwa hal-hal yang dapat merusak sistem dapat dicegah ataupun bila terlanjur terjadi kesalahan-kesalahan dapat diatasi dengan segera.

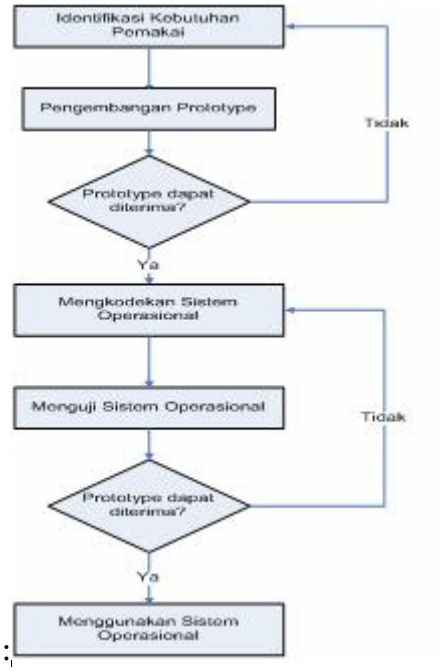
## 2.4 Metode pengembangan sistem

**Prototyping**

Prototyping merupakan modifikasi dari *System Life Cycle* yang lebih rensponsif. Metode ini membangun model kerja kecil dari persyaratan pengguna atau desain yang diusulkan sistem informasi. Maka gambaran lengkap tentang sistemnya pun akan terlihat dari *prototype* yang dibuat. Pengguna dapat melihat pemodelan sistem dari sisi tampilan maupun teknik prosedural yang akan dibangun. Development mencoba efisiensi algoritma, interaksi antara *operating system* dengan *user*.

Tahap identifikasi kebutuhan pemakai merupakan tahap di mana sistem dianalisis untuk mencari kekurangan dan kelebihannya. Setelah mengetahui kelemahannya, kemudian sistem dikembangkan untuk mengatasi kekurangan sistem yang ada. Setelah sistem baru terbentuk, produk tersebut kemudian

dilemparkan kepada pengguna. Jika belum memenuhi kriteria yang diinginkan pengguna, maka harus dilakukan analisis kembali. Tetapi jika kriteria sudah memenuhi keinginan pengguna, maka akan dilakukan pengkodean atas sistem operasi. Kemudian dilakukan pengujian sistem operasi dari *prototype* tersebut. Setelah dilakukan pengujian, *prototype* kembali dilemparkan kepada pengguna. Jika belum memenuhi keinginan pengguna, maka akan dilakukan pengkodean kembali atas sistem operasi dan *prototype* yang telah direvisi. Tetapi jika telah mampu memenuhi keinginan pengguna maka sistem dapat langsung diterapkan dan digunakan. Langkah – langkah *prototyping* digambarkan sebagai berikut :




Gambar 2.1. Model Prosses *Prototype*

[McLeod Jr, Raymond. 2001 : 205]

Metode prototyping memiliki karakteristik sebagai berikut :

1) Resiko tinggi karena tingkat perubahan yang tidak tentu
2) Pertimbangan interaksi pemakai, di mana sistem menyediakan dialog online antara pemakai dengan komputer mikro atau terminal
3) Jumlah pemakai banyak
4) Dibutuhkan penyelesaian yang cepat
5) Perkiraan tahap penggunaan sistem yang pendek
6) Sistem yang inovatif
7) Perilaku pemakai sukar ditebak karena tidak mempunyai pengalaman sebelumnya.

Penulis memilih metode ini karena metode ini sangat fleksibel. Sehingga penulis bebas memulai dari tahap mana saja. Terlebih jika terjadi kesalahan, maka penulis tidak harus memperbaiki awal lagi. Konstruksi sebuah prototype dievaluasi oleh pengguna dan dipakai untuk menyaring kebutuhan pengembangan perangkat keras. Metode ini memungkinkan pengembang mengetahui dengan jelas apa keinginan pengguna sehingga produk yang dihasilkan pun dapat sesuai dengan keinginan pengguna. Meskipun pengguna tekadang tidak sadar bahwa *prototype* yang dibuat bukanlah akhir dari perancangan. Pengembang pun terkadang membuat implementasi dedngan asal – asalan dan teknik *tools* yang tidak optimal pada *prototype.*

Teknik ini pun memilik kekurangan dan kelebihan, yakni :

1) Komunikasi antara analis sistem dengan pemakai baik karena sering berinteraksi

2) Analis dapat bekerja lebih baik dlam menentukan kebutuhan pamakai
3) Pemakai berperan aktif dalam pengembangan sistem
4) Spesialis informasi dan pemakai menghabiskan lebih sedikit waktu dan usaha dalam mengembangkan sistem
5) Penerapan menjadi lebih mudah karena pemakai mengetahui apa yang diharapkannya
6) Menghemat biaya pengembangan sistem
7) Meningkatkan kepuasan pemakai dengan sistem yang dihasilkan
8) Ketergesaan untuk membuat *prototype* mungkin menghasilkan jalan pintas dalam definisi permasalahan, evaluasi alternatif dan dokumentasi
9) Pemakai mungkin sangat tertarik dengan *prototype* tersebut sehingga mereka mengharapkan sesuatu yang tidak realistis dari sistem operasionalnya. Karena itu sulit untuk membuat algoritma menjadi sempurna.
10) Hubungan komputer dengan manusia yang disediakan oleh peralatan *prototyping* tertentu mungkin tidak mencerminkan teknik perancangan yang baik

## 2.5 Metode pendekatan sistem

### Metode object oriented

Object Oriented Design (OOD) menerjemahkan model Object Oriented Analysis (OOA) dari dunia nyata ke dalam model implementasi spesifik yang dapat direalisasikan di dalam perangkat lunak. Proses OOD dapat digambarkan sebagai piramida yang terdiri dari empat lapisan. Lapiasn dasar berfokus pada

desain subsistem yang mengimplementasikan fungsi – fingsi sistem mayor; lapisan kelas mengkhususkan keseluruhan arsitektur objek dan hirarki kelas yang diperlukan untuk mengimplementasikan suatu sistem; Lapisan pesan menunjukkan bagaimana kolaborasi antar objek akan direalisasi; dan lapisan tanggung jawab mengidentifiksikan atribut dan operasi yang menandai masing – masing kelas.

Sasaran OOA adalah mengembangkan sederetan model yang menggambarkan perangkat lunak komputer pada saat perangkat lunak itu bekerja untuk memenuhi serangkaian persyaratan yang ditentukan oleh pelanggan. Analisis terstrusktur mengadopsi pandangan input, proses dan output yang berbeda mengenai persyaratan. Data dipertimbangkan secara terpisah dari proses mentransformasi data.

Banyak metode OOA yang dikembangkan. Maisng – masing memperkenalkan proses untuk analisis produk atau sistem. Serangkaian model yang berkembang keluar dari proses itu dan notasi yang memungkinkan perekayasa perangkat lunak menciptakan masing – masing model dengan cara yang konsisten. Beberapa metode yang ada di antaranya [Presman,688]:

a. Metode Booch

   Metode Booch meliputi proses pengembangan mikro maupun proses pengembangan makro. Tingkat mikro menentukan serangkaian tugas analisis yang diaplikasikan lagi untuk masing 0 masing langkah pada proses makro. Dengan demikian, pendekatan evolusioner dijaga. Metode Booch didukung oleh berbagai piranti otomatis. Outline singkat dari proses pengembangan mikro metode OOA Booch adalah sebagai berikut :

1) Identifikasi kelas dan objek
2) Indentifikasi semantik dari kelas dan objek
3) Identifikasi hubungan di antara kelas dan objek
4) Lakukan sederetan penyaringan
5) Implementasikan kelas dan objek (dalam konteks OOS, hal ini mengimplikasikan perlengkapan kmodel analisis)

b. Metode Coad dan Yourdon

Metode Coad dan Yourdon sering dianggap sebagai salah satu dari metode OOA yang peling mudah untuk dipelajari. Notasi pemodelan relatif sederhana dan pedoman untuk mengembangkan model analisis tersebut jelas. Outline singkat dari proses pengembangan metode OOA Coad dan Yourdon adalah sebagai berikut :

1) Identifikasi objek dengan menggunakan kroteria ‘apa yang dicari’
2) Tentukan struktur generalisasi – spesifikasi
3) Tentukan struktur meseluruhan- bagian
4) Identifikasi subjek (representasi dari komponen subsistem)
5) Tentukan atribut
6) Tentukan pelayanan

c. Metode Jacobson

Metode ini disebut juga OOSE (rekayasa perangkat lunak berorientasi objek) metode Jacobson versi sederhana dari metode objectory sebelumnya, yang juga dikembangkan oelh Jacobson. Metode ini dibedakan dari metode yang lain karena sangat menekankan use case – dekripsi atau skenario yang menggambarkan bagaimana pemakai berinteraksi dengan prosuk atau sistem.

Outline singkat dari proses pengembangan metode OOA Jacobson adalah sebagai berikut :

1) Identifikasi pemakai sistem dan seluruh tanggung jawab mereka
2) Bangun model persyaratan
3) Bangun model analisis

d. Metode Rambough

Rambough dan rekan – rekannya mengembangkan Object Modelling Technique (OMT) menggunakan desain sistem dan desain tingkat objek. Aktivitas analisis menciptakan tiga model : model objek (representasi objek, kelas, hirarki dan hubungan), model dinamis (representasi dari objek dan tingkah laku sistem) dan model fungsional (representasi aliran informasi seperti DFD tingkat tinggi melalui sistem tersebut). Outline singkat dari proses pengembangan OOA Rambough adalah sebagai berikut :

1) Kembangkan pernyataan ruang lingkup masalah
2) Bangun model objek
3) Kembangkan model dinamis
4) Buat model fungsional untuk sistem tersebut

Perlu dicatat bahwa kerja mulai menggabungkan metode Booch dan Rambough. Himpunan resultan dari notasi dan heuristik yang disebut juga Unified Method menggabungkan penekanan data Rambough yang kuat dengan tingkah laku dan tekanan operasional dari Booch.

e. Metode Wirfs-Brock

Metode Wirfs tidak membuat perbedaan yang jelas antara analisis dan tugas desain, malahan mengusulkan proses kontinu mulai dengan penilaian

terhadap spesifikasi pelanggan dan berakhir dengan desai. Outline singkat mengenai tugas yang berhubungan dengan aktivitas Wirfs - Brock adalah sebagai berikut :

1) Evaluasi spesifikasi pelangan
2) Gunakan uraian gramatikal untuk mengekstrak keal calon dari spesifikasi tersebut
3) Kelompokkan kelas dengan tujuan untuk mengidentifikasi superkelas
4) Tentukan tanggung jawab untuk masing – masing kelas
5) Identifikasi hubungan antar kelas
6) Tentukan kolaborasi di antara kelas berdasarkan tanggung jawab
7) Bangun representasi hirarki kelas untuk memperlihatkan hubungan pewarisan
8) Bangunlah grafik kolaborasi untuk sistem

Meskipun terminologi dan langkah proses untuk masing – masing metode OOA ini berbeda, keseluruhan proses OOA benar – benar mirip.

Untuk melakukan OOA perangkat lunak harus melakukan langkah – langkah generik sebagai berikut :

1) Cari persyaratan pelanggan untuk sistem OOA
2) Pilih kelas dan objek dengan menggunakan persyaratan dasar sebagai panduan
3) Identifikasi atribut dan operasi untuk masing objek sistem
4) Tentukan struktur dan hirarki yang mengorganisasi kelas

5) Bangun suatu model hubungan objek
6) Bangun suatu model tingkah laku objek
7) Kaji model analisis OO terhadap use case skenario

Pemrograman OO memperluas model desain ke dalam domain yang dapat dieksekusi. Ahasa pemrograman OO digunakan untuk menerjemahkan kelas, atribut, operasi dan pesan ke dalam suatu bentuk yang dapat dieksekusi oleh sebuah mesin. Akar masalah perangkat lunak mungkin beada dalam terminologi yang paling tradisional untuk menggambarkan industri kita – pemrosesan data Kita belajar bahwa data dan pemrosesan adalah dua hal yang berbeda yang mendasarbagi pasar bisnis. Partisi dapat sangat lebih merusak dari yang kita sadari. OOD memberi kita alat untuk untuk mem-*breakdown* partisi antara data dan proses. Dengan melakukannya, maka kualitas perangkat lunak dapat ditingkatkan.

Data dan operasi Dalam desain berorientasi objek, yang memanipulasi data tersebut dienkapsulasi ke dalam objek. OO juga harus menggambarkan organisasi data spesifik dari atribut serta detail prosedural operasi individu. Hal ini merepresentasikan data dan bagian algoritma juga merupakan penyumbang bagi keseluruhan modularitas.

Sifat OO yang unik ini ada pada kemampuannya untuk membangun empat konsep desain perangkat lunak yang penting, yaitu :

a. Abstraksi

Pada saat kita mempertimbangkan solusi modular terhadap setiap masalah, banyak tingkat abstraksi yang dapat diperoleh. Urutan tertinggi adalah

lingkungan masalah, lalu orientasi yang lebih prosedural, lalu terminologi untuk menyetakan solusi, barulah abstraksi pada tingkat terendah di mana solusi diimplementasikan secara langsung.

Abstraksi data adalah kumpulan data yang bernama yang menggambarkan objek data. Abstraksi kontrol adalah bentuk ketiga dari abstraksi yang dipakau dalam desain perangkat lunak. Seperti abstraksi prosedural dan data, abstraksi kontrol mengimplikasikan suatu mekanisme kontrol program tanpa menentukan detail internal.

b. Penyembunyian informasi

Arsitektur program dikembangkan dengan cara urut menyaring tingkat detail prosedural. Penyaringan sebenarnya adalah proses elaborasi. Penyaringan membuat desainer berelaborasi pada statemen orisinil, yang memberi detail yang lebih banyak banyak lagi pada setiap penyaringan (elaborasi) berurutan terjadi. Abstraksi penyaringan adalah konsep komplementer. Abstransi memungkinkan desainer menentukan prosedur dan data dan masih menekan detail tingkat rendah. Penyaringan membantu desainer untuk mengungkap detail – detail tingkat rendah ketika desain berjalan. Kedua konsep itu membantu desainer dalam menciptakan suatu model desain lengkap jika desain berkembang.

c. independensi fungsional

d. modularitas

Arsitektur perangkat lunak memasukkan modularitas di mana perangkat lunak dibagi ke dalam komponen – komponen bernama dan dapat dipangil terpisah. Meyer dalam bukunya yang disadur kembali oleh Presman [Presman, 410] menyebutkan ada lima kriteria yang memungkinkan kita

mengevaluasi suatu metode desain dengan merujuk pada kemampuannya untuk menentukan sistem modularitas yang efektif, yaitu :

1) Dekomposabilitas modular

Bila metode desain memberikan suatu mekanisme sistematis untuk melakukan dekomposisi terhadap masalah menjadi submasalah – submasalah, maka metode desain akan mengurangi kompleksitas keseluruhan masalah, sehingga dapat mencapai solusi modular yang efektif.

2) Komposabilitas modular

Bila suatu metode desain memungkinkan komponen desain (reusable) yang ada untuk dipasang ke dalam suatu sistem baru, maka metode desain akan menghasilkan sebuah solusi modular yang berulang – ulang.

3) Kemampuan pemahaman modular

Jika sebuah modul dapat dipahami sebagai uit yang berdiri sendiri (tanpa referensi dari modul lain), maka modul akan lebih mudah dibangun dan diubah.

4) Kontinuitas modular

Bila perubahan kecil pada persyaratan sistem menyebabkan perubahan kecil pada modul indvidual dan bukan perubahan sistem secara luas, maka pngaruh dari efek samping yang disebabkan oleh perusahaan dapat diminimalkan.

5) Proteksi modular

Bila terjadi kondisi yang emnyimpang pada modul tersebut, pengaruh dari efek samping yang disebabkan oleh kesalahan akan diminimalkan.

Selama desain subsistem, disinggung empat komponen, yaitu komponen domain masalah, komponen interaksi manusia, komponen manajemen tugas dan komponen manajemen data. Komponen domain masalah mengimplementasikan persyaratan pelanggan bagi aplikasi OO. Komponen yang lain memberikan infrastruktur desain yang memungkinkan aplikasi beroperasi secara efektif. Proses desain objek berfokus pada deskripsi strktur data yang mengimplementasi atribut kelas. Algoritma yang mengimplementasi operasi dan pesan – pesan yang memungkinkan kolaborasi serta hubungan objek

**Unified Modeling Language (UML)**

### 2.5.1 Pengertian Unified Modeling Language (UML)

Pengertian UML menurut Romi Satria Wahono adalah:

“Sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. “

Pengertian UML menurut Software Engineering Laboratory adalah

*“Bahasa untuk menspesifikasikasi, menvisualisasi, membangun dan mendokumentasikan artifcts (bagian dari informasi yang digunakan atau dihasilkan oleh proses pembuatan perangkat lunak, artifact tersebut dapat berupa model, deskripsi atau perangkat lunak) dari sistem perangkat lunak. “*

UML menawarkan sebuah standar untuk merancang model sebuah sistem. UML merupakan pengganti dari metode analisis berorientasi object dan design berorientasi object (OOA&D) yang dimunculkan sekitar akhir tahun 80-an dan awal tahun 90-an

UML merupakan gabungan dari metode Booch, Rumbaught (OMT) dan Jocobson. Tetapi UML ini akan mencakup lebih luas dari OOA dan OOD. Pada pertengahan pengembangan UML dilakukan standarisasi proses dengan OMG (Object Management Group) dengan harapan UML akan menjadi bahasa standar pemodelan pada masa yang akan datang.

UML disebut sebagai bahasa pemodelan buka metode. Kebanyakan metode terdiri paling sedikit prinsip, bahasa pemodelan dan proses. Bahasa pemodelan ( sebian besar grafik) merupakan notasi dari metode yang digunakan untuk mendesain secara cepat.

### 2.6.2 Bagian - bagian dari UML

Bagian–bagian utama dari UML adalah *view, diagaram, model element, dan general mechanism.*

**1. View**

*View* digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda. *View* bukan melihat gratifk, tapi merupakan suatu abstraksi yang berisi sejumlah diagram. Beberapa jenis view dalam UML antarai lain : *use case view, logical viw , component viwm , concurrences view,* dan *deployment view*.

**a. Use case view**

Mendeskripsikan fungsionalitas sistem yang seharusnya dilakukan sesuai yang diinginkan *external actors. Actor* yang berinterakasi dengan sistem dapat berupa user atau sistem lainnya. *View* ini digambarkan dalam *use case diagrams* dan kadang–kadang dengan *activity diagrams.*View ini digunakan terutama untuk

pelanggan , perancang (*designer*) , pengembang *(developer*), dan penguji sistem (*tester*).

**b. Logical view**

Mendeskripsikan bagaimana fungsionalitas dari sistem, struktur statis (*class , object,* dan r*elationship*) dan kolaborasi dinamis yang terjadi ketika *object* mengirim pesan ke *object* lain dalam suatu fungsi tertentu. View ini digambarkan dalam *class diagrams* untuk struktur statis dan dalam *state sequence, collaboration, dan activity diagrams* untuk model dinamisnya, *View* ini digunakan untuk perancang (designer) dan pengembang (developer).

**c. Component view**

Mendiskripsikan implementasi dan ketergantungan modul. Komponen yang merupakan tipe lainnya dari *code module* diperlihatkan dengan struktur dan ketergantungannya juga alokasi sumber daya komponen dan informasi administrative lainnya. *View* ini diganbarkan dalam *component view* dan digunakan untk pengembang (developer).

**d. Concurrency view**

Membagi sistem ke dalam proses dan prosessor. *View* ini digambarkan dalam diagram dinamis (*state, sequence, collaboration,* dan *activity diagrams*) dan diagram implementasi (*component* dan *deployment diagrams*) serta digunakan untk pengembang ( developer), pengintegrasi (integrator), dan penguji (tester).

**e. Deployment view**

Mendeskripsikan fisik dari sistem seperti komputer dan perangkat (*nodes*) dan bagaimana hubungannya dengan lainnya. *View* ini digambarkan dalam

deplovment diagrams dan digunakan untuk pengembang (developer), pengintegrasi (integrator), dan penguji (tester).

## 2. Diagram

Diagram berbentuk grafik yang menunjukan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu *view* tertentu dan ketika digambarkan biasanya dialokasikan untuk *view* tertentu. Adapun jenis diagram antara lain :

### a. Use Case Diagram

Menggambarkan sejumlah *external actors* dan hubungannya ke *use case* yang diberikan oleh sistem. *Use case* adalah deskirpsi fungsi yang disediakan oleh sistem dalam bentuk teks sebagai dokumentasi dari *use case symbo*l namun juga dilakukan dalam *activity diagrams*. *Use case* digambarkan hanya yang dilihat dari luar oleh *actor* (keadaan lingkungan sistem yang dilihat user) dan bukan bagaimana fungsi yang ada di dalam system.

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuag use case mepresentasikan sebuah interaksi antara aktor dengan sistem. Use case merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-create sebuah daftar belanja, dan sebagainya.

### b. Class Diagram

Menggambarkan struktur statis *class* di dalam sistem. *Class* merepresentasikan sesuatu yang ditangani oleh sistem. *Class* dapat berhubungan dengan yang lain melalui berbagai cara : *associated* (terhubunga satu sama lain),

*dependent* (satu *class* tergantung / menggunakan *clas*s yang lain), *speciated* (satu *class* merupakan spesialisasi dari *class* lainnya), atau *package* (grup bersama sebagai satu unit). Sebuah sistem biasanya mempunyai beberapa *class diagram.*

**c. State Diagram**

Menggambarkan semua *state* (kondisi) yang dimiliki oleh suatu *objec*t dari suatu *class* dan keadaan yang menyebabkan *state* berubah.. Kejadian dapat berupa *object* lain yang mengirim pesan. *State class* tidak digambarkan untuk semua *class*, hanya yang mempunyai jumlah *state* yang terdefinisi dengan baik dan kondisi *class* berubah oleh *state* yang berbeda.

**d. Sequence Diagram**

Menggambarkan kolaborasi dinamis antara sejumlah object. Kegunaannya untuk menunjukan rangkaian pesan yang dikirim antar *object* juga interaksi antara *object* sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem.

**e. Collaboration Diagram**

Menggambarkan kolaborasi dinamis seperti *sequence diagrams*. Dalam menunjukan pertukaran pesan, *collaboration diagrams* menggambarkan *object* dan hubungannya (mengacu ke konteks). Jika penekannya pada waktu atau urutan digunakan *sequence diagrams*, tapi jika penekannya pada konteks gunakan *collaboration diagrams.*

**f. Activity Diagram**

Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktivitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktivitas lainnya seperti *user case* atau interaksi.

**g. Component Diagram**

Menggambarkan struktur fisik kode dari kompen. Komponen dapat berupa *source code*, komponen biner, atau *executeable component*. Sebuah komponen berisi informasi tentang *logic class* atau *class* yang diimplementasikan sehingga menbuat pemetaan dari logical view ke *component view*.

**h. Deployment Diagram**

Menggambarkan arsitektur fisik dari perangkat keras dan perangkat lunak sistem, menunjukan hubungan komputer dengan perangkat *( nodes*) satu sama lain dan jenis hubungannya. Di dalam *nodes*, *executeable component* dan *object* yang dialokasikan untuk memperlihatkan unit perangkat lunak yang dieksekusi oleh *node* tertentu dan ketergantungan komponen.

### 2.6.3 Gambaran dari UML

**1. UML sebagai bahasa Pemodelan**

UML merupakan bahasa pemodelan yang memiliki pembendaharaan kata dan cara untuk mempresentasikan secara fokus pada konseptual dan fisik dari suatu sistem. Contoh untuk sistem software yang *intensive* membutuhkan bahasa yang menunjukan pandangan yang berbeda dari arsitektur sistem, ini sama seperti menyusun.mengembangkan *software develpoment life cycle*. Dengan UML akan memberitahukan kita bagaimana untuk membuat dan membaca bentuk model yang baik, tetapi UML tidak dapat membertahukan model apa yang akan dibangun dan kapan akan membangun model tersebut. Ini merupakan aturan dalam *software development prosess.*

2. **UML sebagai bahasa untuk Menggambar kan Sistem (Visualizing)**

UML tidak hanya merupakan simbol grafikal, cukup dengan tiap simbol pada notasi UML merupakan penetapan semantik yang baik. Dengan cara ini, satu pengembang dapat menulis model UML dan pengembang lain atau perangkat yang sama lainnya dapat mengartikan bahwa model tersebut tidak ambigu. Hal ini akan mengurangi *error* yang terjadi karena perbedaan bahasa daalm komuniksi model konseptual dengan model lainnya.

3. **UML sebagai bahasa untuk Menspesifikasi Sistem (Specifying)**

Maksudnya membangun model yang sesuai, tidak ambigu dan lengkap. Pada faktanya UML menunjukan semua spesifikasi keputusan analisis, desain dan implementasi yang penting yang harus dibuat pada saat pengembangan dan penyebaran dari sistem software intensif.

4. **UML sebagai bahasa untuk Membangun Sistem (Constructing)**

UML bukan bahasa pemograman visual, tetapi model UML dapat dikoneksikan secara langsung pada bahasa pemograman visual.

5. **UML sebagai bahasa untuk Pengdokumentasian Sistem (Documenting)**

Maksudnya UML menunjukan dokumentasi dari arsitektur sistem dan detail dari semua UML, hanya memberikan bahasa untuk memperlihatkan permintaan dan untuk tes UML, menyediakan bahasa untuk memodelkan aktivitas dari perencanaan project dan manajemen pelepasan (*release management*).

### 2.6.4 Area Penggunaan UML

UML digunakan paling efektif pada domain seperti :

- Sistem Informasi Perpustakaan
- Sistem Perbankan dan Perekonomian
- Bidang Telekomunikasi
- Bidang Transportasi
- Bidang penerbangan
- Bidang perdagangan
- Bidang Pelayanan Elektronik
- Bidang Pengetahuan
- Bidang pelayanan Berbasis Web Terdistribusi

Namun UML tidak terbatas untuk pemodelan software. Pada faktanya UML banyak untuk memodelkan sistem non software seperti :

- Aliran kerja pada sistem perundangan
- Struktur dan kelakuan dari Sistem Kepedulian Kesehatan Pasien
- Desain hardware dll.

### 2.6.5 Tujuan Pengunaan UML

1. Memodelkan susatu sistem (bukan hanya perangkat lunak) yang menggunakan konsep beorientasi *object*
2. Menciptakan suatu bahasa pemodelan yang dapat digunakan baik oleh manusia maupun mesin.

## 2.7 Komponen-Komponen Yang Terlibat Dalam Use Case Diagram

**1. Actor**

Pada dasarnya *actor* bukanlah bagian dari *use case diagram* , namun untuk dapat terciptanya suatu *use case diagram* diperlukan beberapa *actor* dimana *actor* tersebut mempresentasikan seseorang atau sesuatu (seperti perangkat, sistem lain) yang berinteraksi dengan sistem. Sebuah *actor* mungkin hanya memberikan informasi inputan pada sisstem, hanya menerima infromasi dari sistem atau keduannya menerima dan memberi informasi pada sistem, *actor* hanya berinterkasi dengan use case tetapi tidak memiliki kontrol atas *use case*. *Acttor* digambarkan dengan *stick man.*

Ada beberapa kemungkinan yang menyebabkan *actor* tersebut terkait dengan sistem antara lain :

a. Yang berkepentingan terhadap sistem dimana adanya arus informasi baik yang diterimanya maupun yang dia inputkan ke sistem
b. Orang ataupun pihak yang akan mengelola sistem tersebut
c. *External resource* yang digunkan oleh sistem
d. Sistem lain yang berinteraksi dengan sistem yang akan dibuat

**2. Use Case**

*Use Case* adalah gambaran fungsionalitas dari suatu sistem , sehingga *costumer* atau pengguna sistem paham dan mengerti mengenai kegunaan sistem yang akan di bangun. *Use case* diagram adalah penggambaran sistem dari sudut pandang pengguna sistem tersebut (*user*), sehingga pembuatan *use case* lebih dititikberatkan pada fungsionalitas yang ada pada sistem , bukan berdasarkan alur atau urutan kejadian. Cara menentukan Use Case dalam suatu sistem :

a. Pola perilaku perangkat lunak aplikasi

b. Gambaran tugas dari sebuah *actor*

c. Sistem atau “benda” yang memberikan sesuatu yang berniali kepada *actor*

d. Apa yang dikerjakan oleh suatu perangkat lunak (bukan bagaimana cara mengerjakannya ).

**3. Relasi dalam Use Case**

Ada beberapa relasi yang terdapat pada *use case diagarm* :

a. *Association* , menghubungkan link antar element

b. *Generaliization* , disebut juga *inheritance* (pewarisan), sebuah element pdapat merupakan spesialiasi dari elemen lainnya.

c. *Dependency*, sebuah element bergantung dalam beberapa cara elemen lainnya.

d. *Aggregation*, bentuk assosiation dimana sebuah elemen berisi elemen lainnya

**4. Use Case Diagram**

Adalah gambaran grapichal dari beberapa atau semua actorm use case dan interaksi diantaranya yang memperkenalkan suatu sistem.

## 2.8 Class Diagram

### 2.8.1 Definsi Object dan Class

*Object* adalah gambaran dari entity, baik dunia nyata atau konsep dengan batasan – batasan dan pengertian yang tepat. *Objec*t bisa mewakili sesuatu yang

nyata seperti komputer. Setiap *object* dalam sistem memiliki tiga karakteristik, yaitu *State* (stataus), *Behaviour* (sifat) dan *Identity* (identitas).

Cara mengidentifikasi *object* :

1. Pengelompokan berdasarkan kata / frase benda pada skenario
2. Berdasarkan daftar kategori object , antara lain :
    a. object fisik, contoh : pesawat telepon
    b. spesifikasi /rancangan/deskirpsi, contoh : deskripsi pesawat
    c. tempat , contoh : gudang
    d. transaksi , contoh : penjualan
    e. butir yang terlibat pada transaksi , contoh : barang jualan
    f. peran, contoh : pelanggan
    g. wadah , contoh : pesawat terbang
    h. piranti , contoh PABX
    i. kata benda abstrak, contoh : kecanduan
    j. kejadian, contoh: pendaratan
    k. aturan atau kebijakan, contoh : aturan diskon
    l. catalog atau rujukan , contoh : daftar pelanggan

*Class* adalah deskripsi sekelompok *objec*t dari *property* (atribut), sifat (operasi) relasi antar *object* dan sematik yang umum. *Class* merupakan *template* untuk membent5uk *objex*t. Setiap *objec*t merupakan contoh dari beberapa *class* dan *object* tidak dapt menjadi contoh lebih dari *satu class*. Penamaan *class* menggunakan kata benda tunggal yang merupakan abtraksi yang terbaik. Pada UML , *class* digambarkan dengan segi empat yang dibagi. Bagian atas merupakan

nama dari *class*. Bagian yang tengah merupakan struktur dari *class* (atribut) dan bagian bawah merupakan sifat dari *class* (operasi).

### 2.8.2 Interaction Diagram

**1. Use Case Relization**

Fungsionalitas *use case* direpresentasaikan dengan aliran peristiwa–peristiwa. Skenario digunakan untuk menggambarkan bagaimana u*se case–use case* direalisasikan sebagai interaksi antara *object–object. Use case realization* menggambarkan bagaimana realisasi dari setiap *use case* yang ada pada *use case model*. Untuk menggambarkan bagaimana realasasi dari suatu *use case* dapat menggunakan beberapa diagram, diantaranya adalah *Class Diagram owned by Use Case Realization* serta *Interaction Diagram.*

**2. Interaction Diagram**

Merupakan model yang menjelaskan bagaimana sejumlah *object* bekerja sama dalam beberapa kelakuan. *Interaction Diagram* menerangkan kelakuan dari suatu *use case*. Diagram ini menggambarkan sejumlah *object* dari pesan yang dijalankan antar *object* dengan *use case.* Ketika kita memberikan pesan, aksi yang dihasilkan adalah sebuah pernyataan tereksekusi yang membentuk abstraksi dari prosedur komputasi. Dalam UML, kita dapat memodelkan beberapa jenis aksi , yaitu :

1. Call : memanggil operasi yang ada pada *object*, *object* mungkin mengirim ke dirinya sendiri , menghasilkan pemanggilan lokal dari operasi
2. Return : mengembalikan nilai dari caller

3. Send : mengirimkan sinyal ke object
4. Create : membuat sebuah object
5. Destroy : mematikan *object*, *object* mungkin saja mematikan diri.

Interaction diagram digunakan ketika kita ingin melihat kelakuan dari beberapa *object* dalam *use care* tunggal. Diagram ini baik saat menunjukan kolaborasi diantara *object-object*, namun kurang baik dalam mendefinisikan *behavior*. Ada dua macam *Interaction Diagram* yaitu : *Sequence Diagra*m dan *Collaboration Diagram.*

**3. Sequence Diagram**

Sequence Diagram menggambarkan interaksi antara sejumlah *object* dalam urutan waktu. Kegunaannya untuk menunjukan rangkaian pesan yang dikirim antar *object* juga interaksi antar *object* yang terjadi pada titik tertentu dalam ekseskusi sistem. Dalam UML, *object* pada *diagram sequence* digambarkan dengan segi empat yang berisi nama dari *object* yang digarisbawahi. Pada *object* terdapat 3 cara untuk menamainya yaitu : nama *object* , nama *object* dan *class* serta nama *class.* Dalam *diagram sequence* setiap *object* hanya memiliki garis yang digambarkan garis putus – putus ke bawah. Pesan antar *object* digambarkan dengan anak panah dari objext yang mengirimkan pesan ke *object* yang menerima pesan.

**4. Collaboration Diagram**

Collaboration Diagram merupakan cara alterantif untuk menggambarkan skenario dari sistem. Diagram ini menggambarkan interaksi *object* yang diatur *object* sekelilingnya dan hubungan antara setiap *object* dengan object yang lainnya.

*Collaboration diagram* berisi :

- *Objec*t yang digambarkan dengan segiempat
- Hubungan antara *object* yang digambarkan dengan garis penghubung
- Pesan yang digambarkan dengan teks danb panah dari object yang mengirim pesan ke penerima pesan

**5. Perbedaan Sequence Diagram dengan Collaboration Diagram**

Sequence Diagram memberikan cara untuk melihat skenario dari sistem berdasarkan waktu (apa yang terjadi pertama kali , apa yang terjadi selanjutnya). User akan lebih mudah membaca dan mengerti tipe diagram ini. Karenanya , sangat berguna pada fase analisis awal. Sedangkan *Collaboration Diagram* cenderung untuk memberikan gambaran besar dari skenario selama kolaborasi disusun dari object sekellilingnya dan hubungan antar *object* yang satu dengan lainnya. Diagram ini akan nampak digunakan pada pengembangan tahap desain , ketika kita merancang implemantasi dari hitungan.

## 2.9 Status (state), Behaviour dan Identify

Status dari *object* adalah suatu kondisi yang mungkin ada. Status dari *object* akan berubah setiap waktu dan ditentukan oleh sejumlah *property* (atribut) dengan nilai dari properti, ditambah relasi *objec*t dengan *object* lainnya. Sifat (*Behaviour*) mementukan bagaimana *object* merespon permintaan dari *object* lain dan melambangkan setiap *object* yang dapat dilakukan. Sifat ini diimplementasikan dengan sejumlah operasi untuk *object*. Identitas (*Identity*) artinya setiap *object* yang unik, Pada UML, object digambarkan dengan segiempat dan nama dari *object* diberi garis bawah.

*Rational Objectory Process* menyarankan untuk menemukan *class –class* dalam sistem yang sedang dibangun dengan mencari *class : boundary, control* dan *entity*. Ketiga *stereotype* ini menggambarkan sebuah sudut pandang model *view controller* sehingga membuat analis dapat membagi sistem dengan memisahkan sudut pandang dari domain dari control yang dibutuhkan oleh sistem. Karena proses analisa dan desain adalah iterasi, daftar *class* akan berubah sesuai waktu.*class* awal mungkin tidak akan menjadi *class* yang akan diimplementasikan. Sehingga *candidate class* sering digunakan untuk menggambarkan himpunan awal dari *class* yang ditemukan pada sistem. Untuk merancang *class diagram*, *Rational Unified Process* yang merupakan hasil pengembangan dari *Raitonal Objectary Process* menggunakan *use case realization* yang menggambarkan bagaimana realisasi dari setiap *use casé* yang ada pada *use case* model. Untuk menggambarkan bagaimana realisasi dari suatu *use case* dapat menggunakan beberapa diagram, diantaranya *Class Diagram Owned by* Use Case *Realization* serta *Interaction Diagram.* Untuk menggambarkan *use case realization* di sini akan menggunakan *class diagram owned by use case realization*. Setiap *use case* yang ada di *breakdown*, sehingga akan dapat terlihat entitas–entitas apa saja yang terlibat dalam merelealisasikan sebuah *use case.* Entitas–entitas ini akan menjadi *candidate class* dalam *Class Diagram.*

## 2.10 Attribut dan Operasi pada Class

### 2.10.1 Attribut

Attribut adalah salah satu *property* yang dimiliki oleh *class* yang menggambarkan batasan dari nilai yang dapat dimiliki oleh *peoperty* tersebut. Sebuah class mungkin memiliki beberapa atribut atau tidak memilikinya sama sekali. Sebuah attribut merepresentasikan beberapa *property* dari sesuatu yang kita modelkan, yang dibagi dengan semua *object* dari semua *class* yang ada. Contohnya, setiap tembok memiliki tinggi, lebar dan ketebalan. Attribut dalam implementasinya akan digambarkan sebagai sebuat daftar (*list*) yang diletakan pada kotak dibawah nama *clas*s. Ia seperti halnya nama *class* merupakan teks. Biasanya huruf pertama dari tiap kata merupakan huruf kapital, terkecuali untuk huruf awal. Sebagai contoh : birthDate

### 2.10.2 Operasi

Sebuah operasi adalah sebuah implemantasi dari layanan yang dapat diminta dari beberapa *object* dari *class*, yang mempengaruhi *behaviour*. Dengan kata lain operasi adalah abstraksi dari segala sesutu yang dapat kita lakukan pada sebuah *objec*t dan ia berlaku untuk semua *object* yang terdapat dalam *class* tersebut. *Class* mungkin memiliki beberapa operasi atau tanpa operasi sama sekali. Contohny adalah sebuah *class* “ kotak “ dapat dipindahkan, dperbesar atau di perkecil.. Biasanya (namun tidak selalu), memanggil operasi pada sebuah *object* akan mengubah data atau kondisi dari *object* tersebut.

### 2.10.3 Pengorganisasian atribut dan operasi

Ketika menggambarkan sebuah *class* kita tidak perlu menampilkan seluruh atribut atau operasi. Karena dalam sebagian besar kasus kita tidak dapat menampilkannya dalam sebuah gambar, karena terlalku banyak nya atribut atau operasinya tersebut untuk ditampilkan. Sehingga kita dapat menampilkan hanya sebagian atau bahkan tidak sama sekali atribut dan operasinya.

## 2.11 Relasi dalam object

Semua sistem terdiri dari *class- class* dan *object.* Kelakuan sistem dicapai melalui kerjasama anter *object*, contohnya sorang mahasisea ditambahkan dalam daftar *class.* Jika daftar *class* memperoleh *message* untuk menambhkan mahasiswa. Interaksi antar object disebut *object* relationship. Dua tipe *relationship* yang ditemukan pada saat analisis adalah *assoiciatio*n dan *aggregation.*

1. **Association Relationships**

Association adalah hubungan semantik *bidirectional* diantara *class-class*. Ini bukan aliran data sebagaimana pada pemodelan desain dan analisan terstruktur data diperbolehkan mengalir dari kedua arah. Asosiasi diantara *class–class* artinya ada hubungan antara *object–object* pada c*lass–class* yang berhubungan. Banyakanya *objec*t yang terhubung tergantung dengan *mutliplicity* pada asosisasi.

2. **Aggregation Relationship**

*Aggregation relationship* adlah bentuk khusus dari asosisasi dimana induk terhubung dengan bagian–batgiannya. Notal UML untuk relasi agregasi adalah

sebuah asosiasi dengan diamond putih melekat pada *class* yang menyatakan induk.

**3. Indikator Multiplicity**

Walaupun multiplicit6y ditentukan untuik,class, multiplicit5y menentukan banyaknay object yang terlibat dal;am relasi. Multiplicity, menentukan banyakan object yang terhubung satu engan yang lainnya. Indikator multiplicity terdapat pada masing–masing akhir garis relasi m baik pada asosiasi maupun agregasi. Beberapa contoh multiplicity adalah :

| | |
|---|---|
| 1 | Tepat Satu |
| 0..* | Nol atau lebih |
| 1..* | Satu atau lebih |
| 0..1 | Nol atau Satu 5..8 range 5 s.d. 8 |
| 4..6..9 | ange 4 s.d. 6 dan 9 |

**4. Reflexive Relationship**

*Multiple object* pada *class* yang sama dapat saling berkomunikasi satu dengan yang lainnya,. Hal ini ditunjukan pada *class diagram* sebagai *relexive association* atau aggregation. Penamaan *role* lebih disukai untuk digunakan pada *reflexive relationship* daripada penamaan *association relationship*.

## 2.12 Inheritance

*Inheritance* merupakan kemampuan untuk membuat hierarki yang terdiri atas *class–class*, dimana terdapat strukur dan atau *behaviour* (kelakuan) diantaraa *class-class*. Istilah *superclass* mewarisi semua atribut, operasi dan *relationship*

yang dipunyai oleh semua *superclass*–superclassnya. *Inheritance* disebut juga hierarki isa (adalah sebuah) atau kind–of (sejenis).

**1. Generalization**

*Generalization* menjamin kemmapun untuk membuat *superclass* yang melingkupi struktur dan *behaviour* yang umumnya pada beberapa *class* dibawahnya (*subclass*).

**2. Specialization**

*Specializatio*n menjamin kemampuan untuk membuat *subclass* yang berfungsi untuk menambah atribut dan operasi *superclass.* Operasi pada *superclass* dapat dioverride oleh *subclass* (konsep *polymorphism*). Tetapi, sebuah *sublcass* seharusnya tidak boleh membatasi sebuah operasi yang didefiniskan dalam *superclass*-nya, dengan kata lain *subclass* seharusnya tidak boleh menyediakan lebih sedikit *behaviour* atau struktur daripada *superclass*nya.

## 2.13 Package

Jika sistem hanya memiliki sedikit *class*, kita dapat mengarunya dengan mudah. Seagian besar sistem dibuat dari banyak *class* sehingga kita memerlukan suatu mekanisme untuk mengelompokan bersama untuk memudahkan dalam hal penggunaan, perawatan, dan penggunaan kembali. *Package* adalah kumpulan dari *package* atau *class* yang berelasi. Dengan mengelompokan *class* dalam *package* kita bisa melihat level yang lebih tinggi dari model kita atau kita bisa menggali model dengan leiobh dalam dengan melihat apa yang ada di alam *package*.

### 2.13.1 Package Relationship

Relasi yang digunakan dalam *pacakge relationship* adalah *dependency relationship* . Jika sebuah *package* A tergantung pada *package* B , hal ini beraklibat satu atau lebih *class- class* di *package* A membuai berkomunikasi dengan satu sama lain. *Public class* di *package* B. *Package* A disebut *client package* dan *pakcage* B disebut *supplier pakcage*.

## 2.14 State Transiton Diagram dan Activity Diagram

### 2.14.1 State Transition Diagram

*Use case* dan skenario menyediakan cara untuk menggambarkan kelakuan sistem yakni interaksi antara *object – object* di dalam sistem. Kadang – kadang diperlukan untuk melihat kelakuan di dalam *object*. *State transition diagram* menunjukan *state – state* dari *object* tunggal , *event – event* atau pesan yang menyebabkan transisi dari satu *state* ke *state* yang lain dan *action* yang merupakan hasil dari perubahan sebuat *state. State transition diagram* tidak akan dibuat untuk setiap *class* di *state,. State transition diagram* hanya dibuat untuk *class* yang berkelakuan dinamis. *Interaction diagram* dapat dipelajari untuk menentukan *dynamic object* di sistem, yaitu *object* yang menerima dan mengirim beberapa pesan. *State tranistion* dioagram juga sangat berguna untuk meneliti kelakuan dari sebuah kumpulan *whole class* dan *control class.*

### 2.14.2 States

*State* adalah sebuah kondisi selama kehidupan sebuah *object* ketika *object* memenuhi beberapa kondisi, melakukan beberapa *action* atau menunggu sebuah

*event*. *State* dari sebuah *object* dapat di karakteristikan oleh nilai dari satu atau lebih atribut – atribut dari *class. State- state* dari sebuah *object* ditentukan dengan pengujian / pemeriksaan atribut = atribut dan hubungan – hubungan dari *objec*t.

### 2.14.3 State Transitions

*State Transition* merepresentasikan sebuah perubahan dari state awal ke sebuah state berikutnya (yang mungkin dapat sama dengan *state* awal.) sebuah *actio*n dapat menyertai sebah *state transition*. Ada dua cara untuk membuat transisi sebuah *state- ototmatis* dan *tidak otomtis*. *State transition* yang otomatis terjadi ketikca *activity* dari *state* awal telah lengkatp – tidak ada *event* yang terasosiasi dengan *statae transition* yang belum bernama.. *State transition* yang tidak otomatis disebabkan oleh sebuah *event* ternama (salah satu dari *objec*t atau dari luar sistem). Kedua tipe dari *state transition* dipertimbangkan untuk membuat waktu nol dan tidak dapat di interupsi. Sebuah *state transition* di representasikan oleh sebuah panah yang meunjukan dari *state* awal ke *state* berikutnya.

### 2.14.4 Special State

Ada dua *state* khusus yang ditambahkan di *state transisiton diagram*. Pertama adalah *start state*. Masing – masing diagram harus mempunya satu dan hanya satu *start- state* ketika *object* mulai dibuat.*State* khusus berikutnya ada *stop state*.

### 2.14.5 State Transition Details

Sebuah *State transition* dapat mempunya sebuat *action* dan / atau sebuah kondisi penjaga (*guard condition*) yang terasosiasi dengannya, dan mungkin juga memunculkan sebuah *even*t. Sebuah *acti*on adalah kelakuan yang terjadi ketika *state transition* terjadi. Sebuah *even*t adalah pesan yang dikirm ke *object* lain di sistem. Kondisi penjaga adalah ekspersi *boolean* dari nilai atribut – atribut yang mengijinkan sebuah *state transition* hanya jika kondisinya benar. Kedua *action* dari pengaja adalah kelakuan dari object dan secara tipikal menjadi oeprasi. SEringkali operasi – operasi ini adalah tersendiri- hal itu , merka digunkana hanya oleh *object* dirinya sendiri.

### 2.14.6 State Details

*Action – action* yang mengiringi seluruh *state transition* ke sebuah state mungkin ditempatkan sebagai sebuah *entry action* dalam *state.* Demikian juga, *action – action* yang mengiringi sebluruh *state transition* keluar dari sebuah *state* mungkin ditempatkan sebagai sebuah aksi keluar dalam state. Kelakuan yang terjadi dalam *state* disebut *activity.* Sebuah *activity* memulai ketika *state* dimasukan dan salah satu dari melengkapi atau diinterupsi oleh sebuah *state transition* yang keluar.

## 2.15 Activity Diagram

*Activity Diagram* memodelkan *work flow* proses bisnis dan urutan akitfitas dalam sebuah proses. Diagram ini sangat mirip dengan *flowchart* karena

memodelkan *workflow* dari satu aktivitas ke aktivitas lainnya atau dari aktivitas ke status.

**Elemen – elemen activity diagram :**

1. *Status start*( Mulai ) dan *end* ( akhir)
2. Aktifitas yang merepresentasikan sebuah langkah dalam *workflow*
3. *Transition* menunjukan terjadinya perubahan status aktifitas *(Transition show what sate follows another)*
4. Keputusan yang menunjukan alternatif dalam *workflow*..
5. *Synchronization bars* yang meunjukan subflow parallel. *Syncronization bars* dapat digunakan untuk menunjukan *concurent thread* pada *workflow* proses bisnis
6. *Swimlanes* yang merepresentasikan *role* bisnis yang bertanggung jawab pada aktifitas yang berjalan.

## 2.16 Konsep dasar basis data

Basis data (Database) dapat dibayangkan sebagai sebuah lemari arsip yang ditempatkan secara berurutan untuk memudahkan dalam pengambilan kembali data tersebut. Basis data menunjukkan suatu kumpulan data yang dipakai dalam suati lingkungan perusahaan/ instansi-instansi penerapan basis data dalam sistem informasi disebut sistem basis data (Database System).

### 2.16.1 Pengertian basis data

Basis data (database) adalah suatu pengorganisasian sekumpulan data yang saling terkait sehingga memudahkan aktivitas untuk memperoleh

informasi. Basis data dimaksudkan untuk mengatasi problem pada sistem yang memakai pendekatan berbasis berkas.

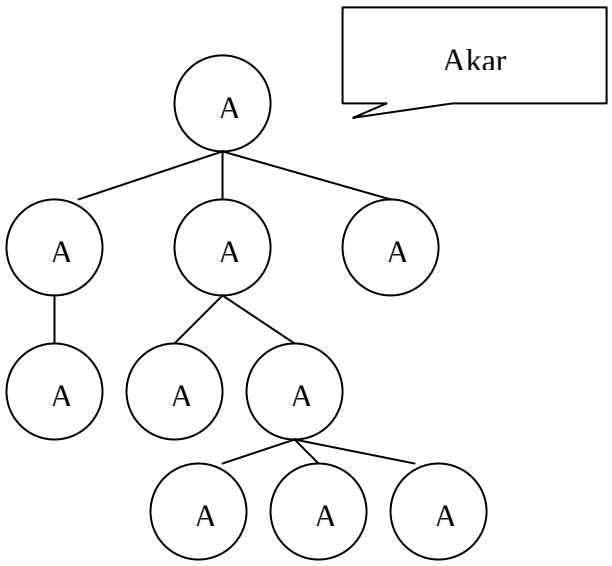
### 2.16.2 Model Basis Data

Yang dimaksud dengan model data adalah sekumpulan konsep terintegrasi yang dipakai untuk menjabarkan data, hubungan antardata, dan kekangan terhadap data yang digunakan untuk menjaga konsistensi. Kadang model data disebut struktur data logis.

Model data yang umum pada saat ini ada 4 macam, yaitu:

1) Model data hierarkis

Model data ini sering dijabarkan dalam bentuk pohon terbalik. Di dalam model ini diknal istilah orang tua dan anak. Masing-nasing berupa suatu simpul dan terdapat hubungan bahwa setiap anak hanya bisa memiliki satu orang tua, sedangkan orang tua dapat memiliki sejumlah anak.

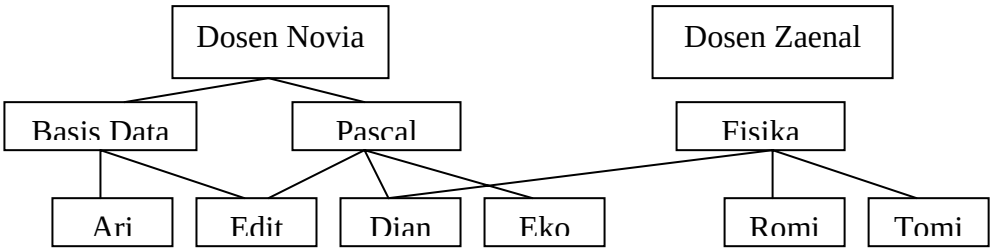



Gambar 2.2 Bentuk model data Hierarkis

(Sumber : Abdul Kadir, Pengenalan Sistem Informasi)

2) Model Data Jaringan

Model data jaringan menyerupai model hierarkis. Namun, ada perbedaan karena model data jaringan : tidak mengenal akar dan setiap anak bisa memiliki lebih dari datu orang tua




Gambar 2.3 Contoh model data jaringan

(Sumber : Abdul Kadir, Pengenalan Sistem Informasi)

3) Model Data Berbasis Objek

Model data berbasis objek adalah model data yang menerapkan teknik pemograman berorientasi objek. Berbeda dengan tiga model yang telah dibahas di depan, model data berbasis objek mengemasdata dan fungsi mengakses data ( metode) ke dalam bentuk objek.

4) Model data Relasional

Model data relasional menggambarkan sekumpulan tabel berdimensi dua ( yang biasa disebut relasi atau tabel) dnegan masing-masing tabel tersusun atas sejumlah baris dan kolom.

## 2.17. Pengujian dan Implementasi

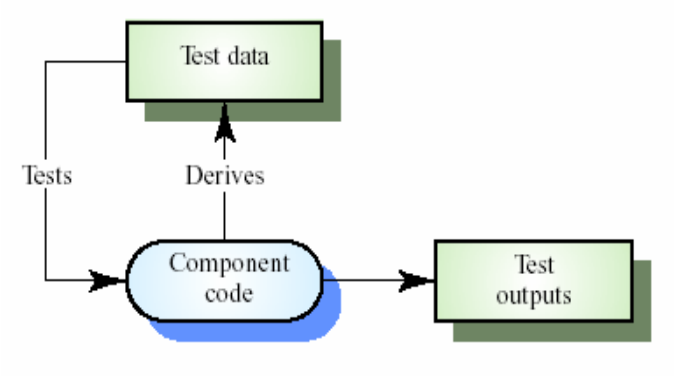
### 2.17.1. Pengujian

Ada dua macam pengujian perangkat lunak yang dapat dilakukan, yaitu metode *white box* dan *blackbox*. *White box testing* dilakukan di awal program, sedangkan *black box testing* baru dilakukan pada tahap berikutnya. Pengujian dilakukan untuk melakukan penilaian tentang keberhasilan program dan menentukan apakah prgram sudah sesuai dengan keinginan *user* atau belum.

*White Box Testing*

*White-box testing* yaitu metode desain *test case* yang menggunakan stuktur kontrol desain posedural unntuk memperoleh *test case*. *Test case* dapat diperoleh dengan:

a. Menjamin bahwa semua *independet path* pada suatu modul telah digunakan minimal satu kali
b. Menggunakan semua keputusan logis pada sisi *true* dan *false*.
c. Mengeksekusi semua *loop* dalam batasannya dan pada batas operasionalnya
d. Mengunakan struktur data internal untuk menjamin validitasnya




Gambar 2.4 . *White box Testing*

(Sumber : www.ilmukomputer.com/ Testing Dan Implementasi)

Pengujian ini harus memenuhi kriteria sebagai berikut :

a. Mengurangi pelaksanaan *test case* untuk mencapai hasil pengujian yang diinginkan
b. *Test case* akan menunjukkan ketidaksesuaian (ketidaksinkronan) beberapa kesalahan, tetapi kurang menunjukkan detil kesalahan

*Software* yang dibangun tidak selalu sempurna. Terkadang ada kesalahan yang baru akan diketahui saat pengujian atau bahkan saat implementasi. Berikut ini kriteria *software* yang "cacat" :

a. Kesalahan logika dan asumsi yang salah berbanding terbalik dengan probabilitas jalur program yang akan dieksekusi. Kecenderungan kesalahan ini terjadi pada tahap desain dan implementasi fungsi, kondisi atau kontrol yang berada di luar pikiran. Kesalahan ini tejadi jika pemrosesan yang rutin sudah dikerjakan dengan baik tetapi pemrosesan yang khusus cenderung diabaikan.
b. Selalu ada keyakinan bahwa *logical path* tidak akan dieksekusi pada basis regular. Kesalahan ini terjadi karena adanya kesalahan asumsi tentang aliran data dan kontrol.
c. Kesalahan tipologis yang merupakan kesalahan yang acak atau random. Perpindahan dari bahasa pemrograman satu ke bahasa pemrograman lain menyebabkan timbulnya kesalahan sintak.

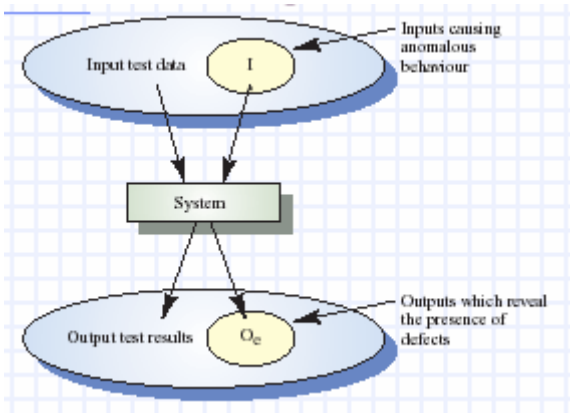
*Path Testing* adalah metode pengujian yang memungkinkan desainer *test case* mengukur kompleksitas logika desain posedural dan menggunakannya

sebagai pedoman untuk menetapkan basis set dari jalur eksekusi. Metode ini menjamin bahwa setiap statement akan dilalui minimal sekali dalam proses pengujian. Tujuannya adalah meyakinkan bahwa himpunan *test case* akan menguji setiap *path* pada suatu program paling sedikit satu kali. Titik awal *path testing* adalah suatu program *flow graph* yang menunjukkan node – node yang menyatakan keputusan program, misalnya : kondisi if – then – else dan busur yang menyatakan alur kontrol. Statement yang memiliki kondisi ada pada node – node dalam *flow graph*. *Path testing* menggambarkan alur kontrol di mana setiap cabang ditunjukkan oleh path yang terpisah sedangkan loop ditunjukkan oleh arrows looping dan kembali ke loop kondisi node.

***Blackbox Testing***

Black-box testing merupakan pendekatan pengujian dimana program dianggap sebagai suatu ‘black-box’ (‘kotak hitam’) Program test case berbasiskan spesifikasi *Test planning* dapat dimulai sejak awal proses pengembangan sistem.

Black-box testing berfokus pada persyaratan fungsional perangkat lunak. Metode ini memungkinkan *software developer* untuk mendapatkan serangkaian kondisi input yang mempergunakan semua persyaratan fungsional program. Black-Box testing bukan alternatif *white-box testing*, namun merupakan pelengkap yang mampu mengungkap kesalahan, jika dibandingkan metode *white-box testing.*




Gambar 2.5. *Black Box Testing*

(Sumber : www.ilmukomputer.com/ Testing Dan Implementasi)

Pengujian black box berusaha menemukan kesalahan dalam kategori :

a. Fungsi – fungsi yang tidak benar atau hilang
b. Kesalahan *interface*
c. Kesalahan dalam struktur data atau akses database eksternal
d. Kesalahan kinerja
e. Inisialisasi dan kesalahan terminasi

*Black-box testing* cenderung diaplikasikan selama tahap akhir pengujian. Pengujian black-box memperhatikan struktur kontrol, sehingga perhatian berfokus pada domain informasi. Black-box testing testing dirancang untuk dapat menjawab pertanyaan berikut :

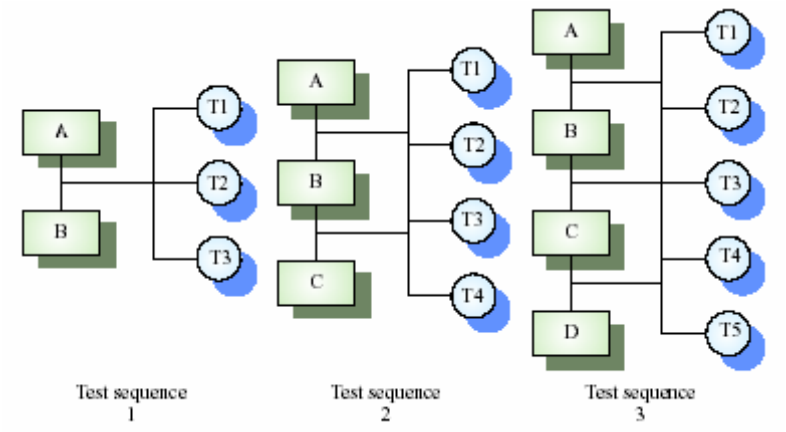
a. Bagaimana validitas fungsional diuji?
b. Kelas input apa yang akan membuat test case menjadi baik?
c. Apakah sistem sensitif tehadap nilai input tertentu?
d. Bagaimana batasan dari suatu data diisolasi?
e. Bagaimana kecepatan dan volume data yang dapat ditolelir oleh sistem?

f. Apa pengaruh kombinasi tertentu dari data terhadap operasi sistem?

*Equivalence partitioning* yaitu metode pengujian *black-box* yang membagi domain input dari suatu program ke dalam kelas data dari mana *test case* dapat dilaksanakan. Input data dan output hasil terdapat di kleas yang berbeda yang sesuai dengan kelas inputnya. Masing-masing kelas *equivalensi partition* diproses dimana program akan memproses anggota kelas-kelas tersebut secara equivalen. *Test cases* dipilih dari masing-masing partisi

Partisi Ekivalensi (*equivalensi partition*) adalah Input data dan output hasil terdapat di kelas yang berbeda yang sesuai dengan kelas inputnya. Masing-masing kelas *equivalensi partition* dimana program akan memproses anggota kelas-kelas tersebut secara *equivale*

*Integration testing* adalah pengujian keseluruhan sistem atau subsistem yang terdiri dari komponen yang terintegrasi. Tes integrasi menggunakan *black box* dengan *test case* ditentukan dai spesifikasi. Kesulitannya adalah menemukan atau melokasikannya. Penggunaan *incremental integration testing* dapat mengurangi masalah tersebut.



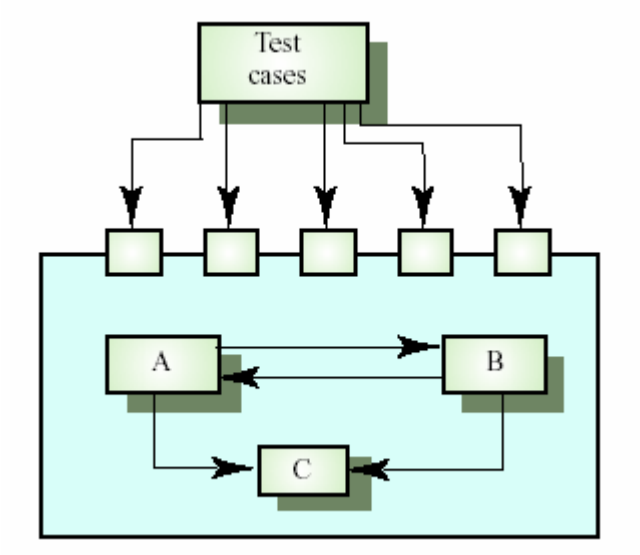

Gambar 2.6 . *Incremental Integration Testing*

(Sumber : www.ilmukomputer.com/ Testing Dan Implementasi)

Ada dua pendekatan *integration testing* yaitu:

a. *Top – Down Testing,* berawal dari level atas sistem dan terintegrasi dengan mengganti masing – masing komponen secara *top- down* dengan suatu *stub* (pogram pendek yang meng-*generate* input ke sub sistem yang diuji).
b. *Bottom – up Testing,* mengintegrasi komponen di level hingga sistem lengkap sudah teruji. Pada prakteknya, kebanyakan test integrasi menggunakan kombinasi kedua strategi pengujian tersebut.

*Interface testing* dilakukan jika modul – modul dan sub sistem terintegrasi dan membentuk sistem yang lebih besar. Tujuannya adalah mendeteksi *fault* terhadap kesalahan interface atau asumsi yang tidak valid tentang inteface tersebut. Pengujian ini sangat penting untuk pengujian pengembangan berorientasi objek yang didefinisikan oleh objek – objeknya.




Gambar 2.7. *Interface Testing*

(Sumber : www.ilmukomputer.com/ Testing Dan Implementasi)

Ada empat tipe interface, yaitu :

a. *Parameter interface,* di mana data dikirim dari satu prosedur ke prosedur lainnya
b. *Shared memory interface,* di mana blok memori di-*share* di antara prosedur – prosedur.
c. *Procedural interface,* di mana sub sistem mengenkapsulasi sekumpulan prosedur – prosedur yang akan dipanggil oleh sub sistem lainnya.
d. *Message passing interface,* di mana sub sistem meminta service dai sub sistem lainnya.

Terdapat beberapa *interface erors*, yaitu :

a. *Interface Misuse,* yaitu jika komponen pemanggil memanggil komponen lainnya dan membuat suatu kesalahan dalam penggunaan interfacenya. Misalnya : parameter dengan urutan yang tidak sesuai.
b. *Interface missunderstanding,* yaitu jika komponen pemanggil salah dalam mengasumsikan komponen *behaviour* yang dipanggil.
c. *Timing errors,* yaitu jika komponen yang memanggil dan yang dipanggil beroperasi pada kecepatan yang berbeda sehingga dimungkinkan mengakses informasi yang tidak *up to date.* Kesalahan terjadi karena *synchronization problem.*

Langkah pertama jika akan melakukan testing interface adalah merancang tes di mana parameter ke prosedur yang dipanggil berada pada nilai batas ekstrim. Perlu diingat bahwa tes menggunakan pointer null. Kemudian perancangan proses dilakukan sehingga komponen yang dites akan *fail.* Bisa juga digunakan *stress testing* pada *message passing. Stress testing* adalah menguji sistem dengan nilai

yang melebihi kapasitas maksimum sistem. Stressing suatu sistem dapat mengurangi kemudahan rusaknya sistem. Sistem seharusnya tidak gagal total. Stess melacak kehilangan *service* yang tidak diduga maupun data yang hilang. Khusus untuk sistem terdistribusi dapat menyebabkan degradasi jaringan sehingga *overload.* Kemudian langkah terakhir adalah memvariasikan urutan di mana komponen diaktifkan pada *shared memory system*.

### 2.17.2. Implementasi Sistem

Menurut WHI [11]

*"Konstruksi sistem adalah pengembangan, instalasi dan pengujian komponen – komponen sistem"*

Penekanannya terletak pada blok pembangunan sistem informasi dari perspektif pembangun sistem. Pemicu fase konstruksi sistem adalah persetujuan spesifikasi dengan desain fisik yang merupakan hasil dai fase desain. Dengan spesifikasi desain yang telah ditentukan, kita dapat mengkonstruksi dan menguji komponen sistem untuk desain tersebut. Akhirnya kita akan membangun sistem fungsional. Sistem fungsional dapat diimplementasikan atau dikirim sebagai sistem operasional.

Tujuan fase konstuksi adalah untuk membangun dan menguji sebuah sistem fungsional yang memenuuhi persyaatan bisnis dan desain dan untuk mengimplementasi inteface antara sistem baru dengan sistem produksi yang telah ada. Pemrograman biasanya dikenal sebagai aspek utama dari fase konstruksi. Akan tetapi deengan kecenderungan solusi sistem yang melibatkan perolehan dan

pembelian paket perangkat lunak, maka implementasi dan integrasi komponen – komponen perangkat lunak menjadi suatu aspek visibel yang beimbang dan umum pada fase konstruksi. Berikut ini tugas fase konstruksi :

a. Membangun dan mengetes jaringan. Fase ini melibatkan analis, desainer dan pembangun sistem. Desainer jaringan dan administrator jaringan memikul tanggung jawab utama untuk menyelesaikan tugas ini. Desainer jaringan adalah seorang yang ahli dalam mendesain LAN dan WAN beserta konektivitasnya. Administrator memiliki keahlian untuk membangun dan menguji teknologi jaringan sistem. Ia juga bertanggung jawab terhadap keamanan sistem. Analis dapat dilibatkan, sebagai fasilitator dan memastikan bahwa persyaratan bisnis tidak dikompromikan dengan solusi jaringan.
b. Membangun dan mengetes database. Tugs ini melibatkan pengguna, analis, desainer dan pembangun sistem. Pengguna sistem memberkan atau menyetujui data pengujian untuk digunakan dalam database. Jika databse yang dibangun bersifat nonkorporat dan berorientasi aplikasi, maka tugas tersebut sering diselesaikan oleh analis sistem. Jika sebaliknya, maka analis sistem harus memastikan terpenuhinya persyaratan bisnis. Desainer database dapat melibatkan programer database untuk membuat dan mempopulasikan database awal. Administrator bertugas memperbaiki performa database, menambah kontrol keamanan dan menyediakan *backup dan recovery*.
c. Menginstal dan mengetes paket perangkat lunak baru. Input utama untuk perangkat lunak ini adalah paket perangkat lunak baru dan dokumentasi

yang diterima dai vendor sistem. Programer aplikasi akan menyelesaikan instalasi dan pengujian paket tersebut sesuai persyaratan integrasi dan dokumentasi program yang dibuat selama desain sistem. Hasil utama tugas ini adalah paket perangkat lunak yang telah diinstal dan diuji dan tersedia dalam libray perangkat lunak. Semua spesifikasi perangkat lunak yang telah dimodifikasi dan persyaratan integrasi baru yang penting didokumentasikan dan tersedia dalam proyek untuk melengkapi sejarah dan befungsi sebagai referensi di masa mendatang.

d. Menulis dan mengetes program baru Prototype termasuk dalam bagian spesifikasi desain teknik untuk menyempurnakan konstruksi dan implementasi sistem. Namun prototype jarang ada yang berjalan dengan sempurna. Karena itu, dibutuhkan pengujian. Ada tiga tingkatan pengujian yang dilakukan, yaitu : *stub testing, unit test (program test) dan system test. Stub test* adalah pengujian yang dilakukan pada satu subset dari sebuah program. *Program test* adalah pengujian yang dilakukan pada suatu program keseluruhan. *System test* adalah pengujian yang dilakukan pada sistem keseluruhan.Setelah pengujian sistem selesai dan dinilai berhasil, kita dapat melangkah ke implementasi sistem.

Menurut WHI [10]

> *"Implementasi sistem adalah instalasi dan pengiriman sistem keseluruhan ke produksi."*

Setiap perubahan sistem akan memerlukan transisi dai sistem lama ke sistem yang baru. Implementasi akan membantu mengatasi masalah *start up* yang terjadi da membawa sistem produksi kepada operasi.

Hasil dari fase implementasi adalah sistem operasional yang akan masuk ke tahap *operation and support* dari siklus hidupnya. Fase ini mempertimbangkan blok pembangun yang sama seperti waktu konstruksi. Tugas fase implementasi adalah sebagai berikut :

a. Menguji Sistem. Setelah perangkat lunak dan program diinstal dan diuji maka harus diadakan tes final terhadap sistem. Analis sistem memfasilitasi penyelesaian tugas ini dan mengkomunikasikan berbagai hal dan masalah dengan anggota tim proyek. Pemilik dan pengguna sistem memiliki wewenang apakah sistem berjalan baik atau tidak. Pembangun sistem memberikan spesialisasi. Sistem tes mungkin akan menghasilkan modifikasi yanng diperlukan oleh program. Iterasi ini akan dilanjutkan sampai pengujian sistem dianggap berhasil.
b. Menyiapkan rencana konversi. Dengan menggunakan spesifikasi desain untuk sistem baru, analis membuat rencana detil konversi. Rencana ini mengidentifikasi database yang harus diinstal, pelatihan pengguna akhir dan dokumentasi yang harus dikembangkan dan strategi yang dapat mengkonversi sistem lama ke sistem yang baru. Rencana konversi dapat melibatkan salah satu dari strategi instalasi seperti di bawah ini :
    1) *Abrupt cut – over*. Pada tanggal tertentu, sistem lama akan berakhir dan digantikan oleh sistem baru. Tidak ada biaya trransisi.

2) *Parallel conversion.* Sistem lama dan baru dioperasikan untuk beberapa periode. Hal ini digunakan untuk memastikan seluruh masalah utama pada sistem baru telah diatasi sebelum sistem lama dibuang.
3) *Location convertion.* Ketika sistem yang sama akan digunakan pada beberapa lokasi geografis, sistem biasanya diubah pada suatu lokasi terlebih dahulu.
4) *Staged conversion. Staged conversion* merupakan variasi pada *abrupt dan parallel conversion* yang didasarkan pada konsep versi yang telah diperkenalkan sebelumnya.
5) *Veification testing.* Yaitu menjalankan sistem pada lingkungan yang tersimulasi dengan menggunakan data yang tersimulasi. Tugas utama pengujian simulasi adalah mencari kesalahan dan kelalaian yang berkenaan dengan spesifikasi pengguna akhir dan desain yang telah ditentukan.
6) *Validation testing.* Item yang diuji selama validasi yaitu : performansi sistem, *peak workload processing performance* (apakah sistem dapat mengatasi beban kerja selama periode pemrosesan puncak), *human engineering test* (apakah sistem mudah dipelajari), *methods dan procedure test* (modifikasi prosedur yang janggal dan tidak efisien menurut pengguna akhir) juga *backup dan recovery testing* (memastikan bahwa data telah benar – benar dapat diperbaiki).

7) *Audit testing.* Menyatakan bahwa sistem bebas dari kesalahan dan siap untuk ditempatkan pada operasi.

c. Menginstal database. Pembangun sistem memegang peran utama dalam aktivitas ini. Programer aplikasi akan membuat program aplikasi untuk mengekstrak data dari database yang sudah ada dan mempopulasikan database bau. Analis sistem akan mengukur database dalam perkiraan waktu yang dibutuhkan selama proses penginstalan. Untuk mempopulasikan database diperlukan program khusus. Hasil utma tugas ini adalah data lama yang sudah direstrukturisasi yang telah dipopulasikan di dalam database untuk sistem baru.

d. Melatih para pengguna. Sistem diperuntukkan bagi pengguna. Mereka harus dilatih untuk mengurangi kemungkinan terjadinya *huuman error* ketika program digunakan nantinya. Analis sistem akan memberikan dokumentasi dan pelatihan yang dibutuhkan pengguna sistem untuk dapat menggunakan sistem baru dengan tepat. Hasil utamanya adalah pelatihan dan dokumentasi pengguna.

e. Beralih ke sistem baru. Pada konversi dari sistem lama ke sistem yang baru, manajer proyek mengawasi tahap tersebut. Pemilik sistem memberikan umpan balik berkenaan dengan sistem baru. Analis, desainer dan pembangun sistem akan menilai umpan balik tentang penggunaan aktual dari sistem tersebut. Input kunci aktivitas ini adalah rencana konversi yang telah dibuat pada fase implementasi sebelumnya. Hasil utamanya adalah sistem operasional yang diletakkan dalam produksi dalam bisnis tersebut.

## 2.18 Arsitektur aplikasi

### 2.18.1 Pengertian Jaringan Komputer

Jaringan komputer ( computer network) atau sering disingkat jaringan saja adalah hubungan dua buah simpul (umunya berupa komputer) atau lebih yang tujuan utamanya adalah untuk melakukan pertukaran data. Bentuk koneksinya tidak harus melalui kawat saja melainkan dapat menggunakan serat optik, gelombang mikro, atau bahkan satelit komunikasi.

### 2.18.2 Jenis-jenis Jaringan Komputer

Ditinjau dari rentang geografis yang dicakup oleh suatu jaringan, jaringan biasa dibagi menjadi 3 macam yaitu :

1) Local Area Network (LAN)

LAN adalah jaringan komputer yang mencakup area dalam satu ruang, satugedung, atau beberapa gedung yang berdekata. Contoh: jaringan satu kampus. LAN menggunakan media transmisi berupa kabel, apabila tidak menggunakan kabel dinamakan *wireless* LAN atau LAN tanpa kabel. Kecepatan LAN berkisar dari 10 Mbps sampai 1 Gbps.

2) Metropolitan Area Network (MAN)

MAN adalah jaringan yang emncakup area satu kota atau dengan rentang sekitar 10-45 Km. Jaringan yang menghubungkan beberapa bank yang terletak dalam satu kota ataukamous yang tersebar dalam beberapa lokasi tergolong termasuk sebagai MAN. Jaringan ini pada umumnya menggunakan media transmisi dengan mikrogelombang atau gelombang radio.

3) Wide Area Network (WAN)

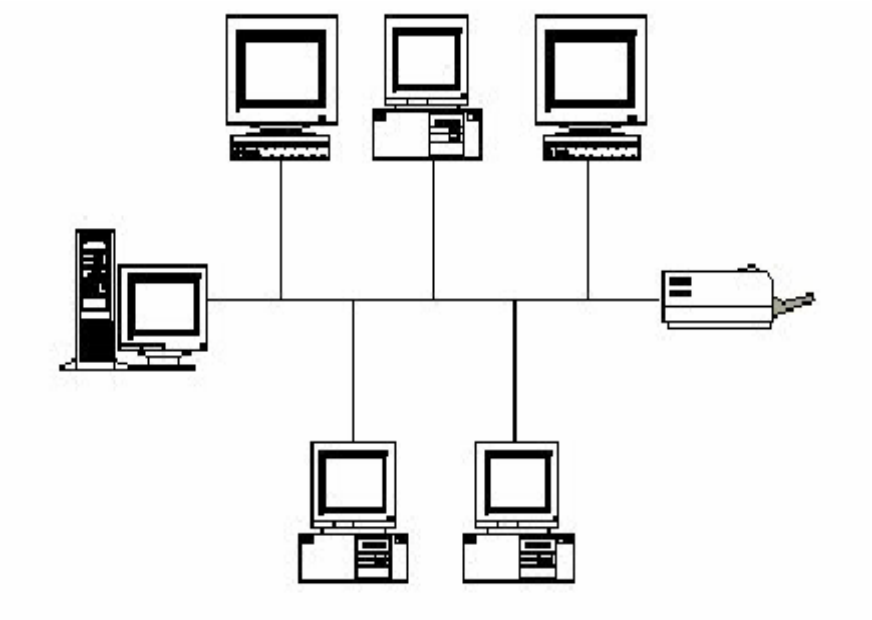
Jaringan yang mencakup antarkota, antarprovinsi. Antarnegara dan bahkan antarbenua disebut dengan WAN. Misalnya jaringan yang menghubungkan ATM, Internet.

### 2.18.3 Topologi Jaringan Komputer

Topologi jaringan menyatakan sususnan komputer secara fisik dalam suatu jaringan. Secara garis besar dapat dibagi menjadi tiga yaitu:

1) Topologi Bus

Pada Topologi ini semua simpul ( umumnya komputer) dihubungkan melalui kabel yang disebut bus. Kabel yang digunakan adalah kabel koaksial.



Gambar 2.8. Topologi Bus

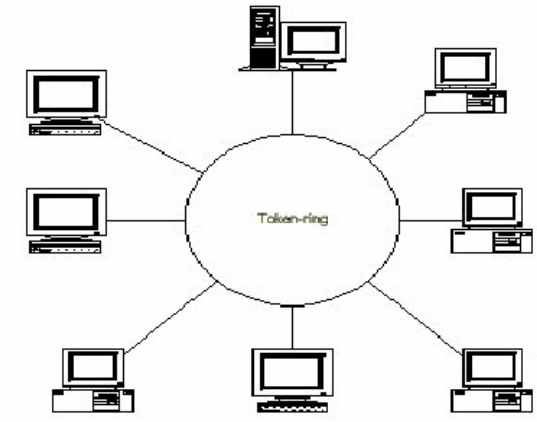
(Sumber : Andri Kristanto. 2003. Jaringan Komputer)

Tabel 2.1. Keuntungan dan Kerugian Topologi Bus

| Keuntungan | Kerugian |
|---|---|
| Instalasi mudah | Jika kabel utama putus maka semua komputer tidak bisa saling berhubungan |
| Biaya murah | Jika kabel sangat panjang, akan sulit bila mencari penyebab kerusakan. |
| Mudah dikembangkan | Jika banyak komputer yang aktif akan sering terjadi tabrakan sehingga kecepatan pengiriman data menjadi pelan. |

2) Topologi Cincin (Ring)

Topologio cincin mirip dengan topologi bus. Informasi dikirim oleh sebuah komputer akan dilewatkan ke media transmisi, melewati satu komputer ke komputer berikutnya.



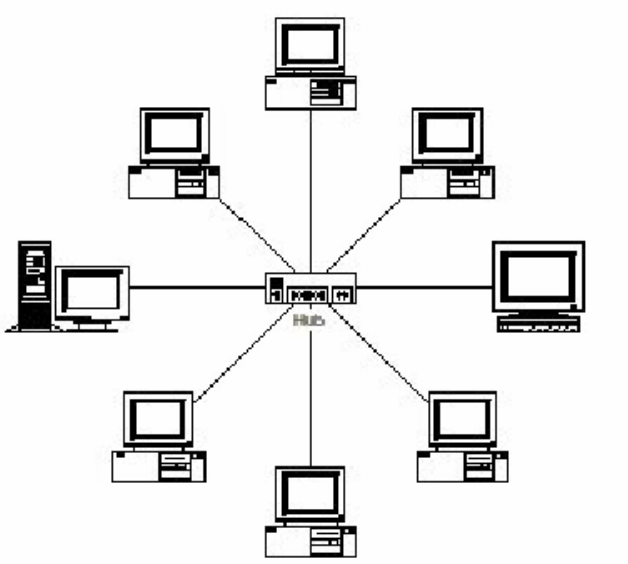

Gambar 2.9. Topologi Cincin

(Sumber : Andri Kristanto. 2003. Jaringan Komputer)

Kelemahan topologi cincin terletak pada kegagalan salah satu simpul. Jika ada satu saja simpul yang mengalami kegagalan, maka semua hubungan terputus. Pada topologi bus, kegagalan pada simpul ( bukan pada bus) tidak mempengaruhi simpul yang lain. Topologi ini biasa digunakan pada LAN.

3) Topologi Bintang

Pada Topologi ini terdapat komponen yang bertindak sebagai pusat terkontrol. Semua simpul yang hendak berkomunikasi selalu melalui pusat pengontrol tersebut. Pusat pengontrol berupa hub atau switch. Topologi ini bisa digunakan untuk LAN,MAN,WAN.



Gambar 2.10. Topologi Bintang

(Sumber : Andri Kristanto. 2003. Jaringan Komputer)

Kelebihan topologi bintang sebagai berikut :

A. Mudah dikelola dan dihubungkan

B. Kegagalan pada sebuah komputer tidak berpengaruh pada kegagalan seluruh jaringan.

Kelemahan topologi bintang sebagai berikut :

A. Kegagalan pada pusat pengontrol akan menyebabkan kegagalan jaringan secara keseluruhan.
B. Jika pusat pengomtrol berupa hub ( bukan berupa switch). Kecepatan trasmisi menjadi lambat.

### 2.18.4 Manfaat Jaringan Komputer

Manfaat terbentuknya jaringan komputer adalah sebagai berikut :

a. *Sharing resource*, bertujuan agar seluruh program, pendataan atau peripheral lainnya dapat dimanfaatkan oleh setiap orang yang ada pada jaringan tanpa terpengaruh oleh lokasi, daerah maupun pengaruh dari pemakai.
b. Komunikasi antar pengguna, baik untuk teleconference maupun mengirim pesan atau informasi penting lainnya. Dengan jaringan komputer, dua orang atau lebih yang jaraknya sangat jauh akan lebih mudah bekerja sama.
c. Integrasi data yang dapat mencegah ketergantungan pada komputer pusat. Setiap proses data tidak harus dilakukan pada satu komputer saja, tapi dapat diditribusikan sehingga pemakai lebih mudah memperoleh dan mengolah data setiap saat.
d. Pengembangan dan pemeliharaan peralatan yang dapat menghemat biaya.
e. Keamanan data yang memberikan perlindungan terhadap data.
f. Sumber daya yang digunakan secara bersama – sama memungkinkan pemakai untuk mendapatkan informasi terkini.

g. Mengurangi ketergantungan pada satu penjual, karena perangkat jaringan pun kini sudah lebih fleksibel.

## 2.19 Internet

Internet merupakan contoh jaringan terbesar yang menghubungkan jutaan komputer yang tersebar di seluruh penjuru dunia dan tidak terikat pada satu organisasipun. Siapa saja dapat bergabung pada internet. Dengan menggunakan jaringan ini, sebuah organisasi dapat melakukan pertukaran informasi secara internal ataupun melakukan pertukaran informasi secara eksternal dengan organisasi-organisasi yang lain.

Secara harafiah, internet (kependekan daripada perkataan 'inter-network') ialah rangkaian komputer yang berhubung menerusi beberapa rangkaian. Manakala Internet (huruf 'I' besar) ialah sistem komputer umum, yang berhubung secara global dan menggunakan TCP/IP sebagai protokol pertukaran paket (packet switching communication protocol). Rangkaian internet yang terbesar dinamakan Internet. Cara menghubungkan rangkaian dengan kaedah ini dinamakan internetworking.

## 2.20 Konsep dasar WWW (World Wide Web)

Sistem pengaksesan informasi dalam internet yang paling terkenal adalah World Wide Web. World Wide Web ("WWW", atau singkatnya "Web") adalah suatu ruang informasi di mana sumber-sumber daya yang berguna diidentifikasi oleh pengenal global yang disebut Uniform Resource Identifier (URI). WWW sering dianggap sama dengan Internet secara keseluruhan, walaupun sebenarnya

ia hanyalah bagian daripadanya. Tujuan awal diciptakannya WWW adalah untuk menciptakan mdia yang mudah untuk berbagi informasi diantara para fisikawan dan ilmuwan.

World Wide Web adalah aplikasi yang paling menarik di internet. Aplikasi ini sangat penting dan banyak sekali digunakan. Drai WWW ini kita bisa mendapatkan informasi tidak hanya teks tetapi juga gambar maupun multimedia. Informasi yang diletakkan di WWW disebut dengan homepage dan setiap homepage mempunyai alamat masing-masing.

## 2.21 HTTP

HTTP (HyperText Transfer Protocol) adalah protokol yang dipergunakan untuk mentransfer dokumen dalam World Wide Web (WWW). Protokol ini adalah protokol ringan, tidak berstatus dan generik yang dapat dipergunakan berbagai macam tipe dokumen. HTTP adalah sebuah protokol meminta/menjawab antara client dan server.

## 2.22 Perangkat lunak penunjang

Dalam pembuatan halaman web kita membutuhkan perangkat lunak sebagai penunjang pembuatannya baik sebagai teks editor maupun sebagai bahasa pemogramannya

### 2.22.1 PHP dan MySQL

PHP merupakan script untuk pemograman script web server- side, script yang membuat dokumen HTML secara *on the fly*, dokumen HTML yang dihasilkan dari suatu aplikasi bukan dokumen HTML yang dibuat dengan editor

teks atau editor HTML. Dengan menggunakan PHP maka maintenance suatu situs web menjadi lebih muah. Proses update dapat dilakukan dengan menggunakan aplikasi yang dibuat dengan menggunakan script PHP.

**Mysql**

SQL ( Structured Query Language ) adalah bahasa standart yang digunakan untuk mengakses server database . Semenjak tahun 70-an bahasa ini telah dikembangkan oleh IBM, yang kemudian diikuti dengan adanya Oracle, Informix dan Sybase. Dengan menggunakan SQL, proses akses database menjadi lebih user-friendly dibandingkan dengan misalnya Base data

MySQL adalah sebuah server database SQL multiuser dan multi-threaded. SQL sendiri adalah salah satu bahasa database yang paling populer di dunia. Implementasi program server database ini adalah program daemon 'mysqld' dan beberapa program lain serta beberapa pustaka.

### 2.22.2 Macromedia Dreamweaver

Macromedia Dreamweaver adalah sebuah HTML editor profesional untuk mendesain secara visual dan mengelola situs atau halaman web.

Dreamweaver merupakan software utama yang digunakan oleh web desainer maupun web programmer dalam mengembangkan suatu situs web. Hal ini disebabkan ruang kerja, faslitas, dan kemampuan Dreamweaver yang mampu meningkatkan produktivitas dan efektivitas dalam desain maupun membangun suatu situs web. Dreamweaver juga dilengkapi dengan fasilitas untuk manajemen situs yangcukup lengkap.

### 2.22.3 Rational Rose

Rational Rose adalah kakas (tools) pemodelan visual untuk pengembangan sistem berbasis objek yang sangat handal untuk digunakan sebagai bantuan bagi para pengembang dalam melakukan analisis dan perancangan sistem. Rational Rose digunkan untuk melakukan pemodelan sistem sebelum pengembang menulis kode-kode dalam bahasa pemograman tertentu. Rational Rose mendukung pemodelan bisnis, yang membantu para pengembang untuk memahami sistem dengan cara pengembang membuat diagram use case untuk melihat fungsionalitas sistem secara keseluruhan sesuai dengan harapan dan keinginan pengguna.

Rational Rose merupakan salah satu software yang paling banyak digunakan untuk melakukan design software melalui pendekatan UML (Unified Modelling Language). Rational Rose merupakan software yang menyediakan banyak fungsi - fungsi seperti : design proses, generate code, reverse engineering, serta banyak fungsi-fungsi yang lain.

Rational Rose merupakan tool yang sangat mudah karena sudah menyediakan contoh-contoh design dari beberapa software. Sebelum menggunakannya, terlebih dahulu pahamilah tentang UML ( Unified Modelling Language ).

Rational Unified Proses (RUP ) merupakan pilihan yang tepat untuk membuat suatu design yang lengkap dengan cara mudah karena model ini merupakan template,sehingga kita dapat langsung melakukan perubahan dengan mengganti use case,actors,class diagram yang

telah ada pada template RUP.

Use case diagram menggambarkan interaksi antara actor dengan proses atau sistem yang

dibuat. Use case diagram mempunyai beberapa bagian penting seperti: Actor, Use Case,Undirectional Association , Generalizati on.

a. Actor merupakan bagian dari use case yang bertindak sebagai subjek ( pelaku) dalam suatu proses.
b. *Use case* adalah proses-proses yang terjadi dalam suatu software.Use case juga menggambarkan apa yang sedang dilakukan oleh seorang Actor.
c. Relasi, menggambarkan hubungan antara actor dan use case.relasi-relasi tersebut dapat dibagi menjadi:
    1) Undirectional Association
    2) Generalization
    3) Dependency

Class diagram menggambarkan interaksi antar class serta atribut-atribut yang melekat pada class tersebut. Sebelum membuat class diagram, terlebih dahulu harus dibuat class-class yang diperlukan untuk sistem tersebut.

*Sequence diagram* merupakan interaksi antara objek-objek dalam suatu sistem dan terjadi komunikasi yang berupa pesan(message) serta parameter waktu.

*Collaboration diagram* hampir sama dengan sequence diagram tetapi berbeda pada objek

yang di titik tekankan , collaboration lebih menekankan pada pemunculan objek itu sendiri. Sedangkan sequence diagram lebih pada penyampaian message dengan parameter waktu. Untuk membuat collaboration diagram dapat secara otomatis dibuat jika sudah ada sequence diagramnya.untuk membuat

collaboration diagram klik browse pada menu kemudian create collaboration diagram, atau dengan menekan tombol F5 pada keyboard.

*Activity diagram* menggambarkan proses-proses yang terjadi mulai aktivitas dimulai sampai aktivitas berhenti.Activity diagram ini mirip dengan flowchart diagram. Untuk membuat activity diagram klik kanan pada design model kemudian klik **new activity diagram.**

*Statechart Diagram* merupakan diagram yang menggambarkan perubahan state dari state yang satu ke state lainnya. Untuk membuat statechart diagram hampir sama dengan membuat activity diagram yaitu:klik kanan pada Package design model kemudian pilih **New Statechart Diagram**.

Salah satu fasilitas dari Rational Rose adalah menyediakan suatu database diagram serta menyediakan generate ke script SQL

## 2.23 Konsep yang dianalisis / dirancang

### 2.23.1 Sistem Informasi Pemesanan kamar

Berdasarkan penjelasan sistem, informasi, dan sistem informasi maka beberapa pengertian yang berhubungan dengan judul skripsi antara lain :

#### 2.23.1.1 Pengertian pemesanan

Proses memilih dan menentukan barang atau jasa yang akan dibeli sebelum dilakukan persetujuan penjualan.

#### 2.23.1.2 Pengertian kamar

Ruangan yang disewakan

#### 2.23.1.3 Pengertian pemesanan kamar

Proses memilih, menentukan dan memesan suatu fasilitas yang disediakan oleh suatu usaha penginapan yang akan dipakai oleh tamu sebelum dilakukannya persetujuan penjualan kamar.

**2.23.1.4 Pengertian Hotel secara umum**

Hotel adalah suatu usaha yang menggunakan suatu bangunan atau bagian dari bangunan daripadanya yang khusus disediakan, dimana setiap orang dapat menginap dan makan serta memperoleh pelayanan dan fasilitas lainnya dengan pembayaran (mempunyai restoran yang berada di bawah manajemen hotel tersebut). Apabila tidak memenuhi persyaratan seperti tersebut di atas dikategorikan sebagai "penginapan". Hotel juga merupakan salah satu usaha komersial dalam sektor pariwisata di Indonesia yang juga memperdampakkan perkembangan pariwisata yang berpotensi dan yang perlu dikembangkan.

Hotel berbintang adalah hotel yang berdasarkan penelitian tim penilai dari Direktorat Jenderal Pariwisata telah memenuhi persyaratan yang telah ditentukan yaitu:

a. Persyaratan fisik : meliputi lokasi hotel, kondisi bangunan dan sebagainya.
b. Bentuk pelayanan yang diberikan (service)
c. Kualifikasi tenaga kerja yang meliputi pendidikan, kesejahteraan karyawan dan sebagainya.
d. Fasilitas Olahrada dan rekreasi lainnya yang tersedia seperti lapangan tenis, kolam renang dan sebagainya.
e. Jumlah kamar yang tersedia.