`BAB II

TINJAUAN PUSTAKA

2.1 Profil Unikom

Profil Unikom berisi penjelasan yang memaparkan tentang sejarah, logo, visi, misi, tujuan, dan struktur organisasi Unikom.

2.1.1 Sejarah Unikom

Universitas Komputer Indonesia (UNIKOM) secara resmi berdiri pada hari Selasa, tanggal 8 Agustus 2000 berdasarkan Surat Keputusan Menteri Pendidikan Nasional nomor 126/D/0/2000.

Awalnya dimulai pada bulan Juli tahun 1994 ketika didirikan Lembaga Pendidikan Komputer Indonesia Jerman, disingkat LPKIG, bertempat di jalan Dipati Ukur 102 Bandung. Dengan 1 ruang kelas berkapasitas 50 orang dan 1 laboratorium komputer dengan 25 unit komputer, Lembaga ini membuka program pendidikan 1 tahun dengan 5 program studi yaitu Ahli Komputer Aplikasi Bisnis, Ahli Komputer Keuangan & Perbankan, Ahli Komputer Akuntansi & Perpajakan, Ahli Komputer Manajemen & Pemasaran dan Sekretaris Eksekutif. Jumlah peserta pendidikan pada tahun pertama ini sebanyak 233 siswa.

Pada tahun kedua, 1995, dibuka jenjang pendidikan 3 tahun untuk memenuhi animo siswa tahun pertama yang ingin memperdalam ilmunya, disamping pemikiran jangka panjang pengembangan institusi. Pada tahun ini juga dibuka program studi baru, meliputi : Ahli Komputer Teknik Informatika, Ahli

Komputer Manajemen Informatika dan Sekretaris Eksekutif. Ruang kelas ditambah menjadi 2 buah dan laboratorium komputer menjadi 2 buah dengan jumlah siswa sebanyak 457 orang.

Pada tahun ketiga, 1996, dilakukan penambahan gedung kuliah baru bertempat di jalan Dipati Ukur 116 (gedung FISIP sekarang), sekaligus pemindahan pusat administrasi dan perkantoran. Digedung baru ini dilakukan penambahan 1(satu) Lab. Komputer, 5(lima) Ruang Kuliah, Ruang Dosen dan Ruang Kemahasiswaan. Jumlah siswa dari tahun 1996 hingga tahun 1998 bertambah dari 632 orang menjadi 1184 orang.

Pada tahun kelima, 1998, dimulai pembangunan Kampus baru (Gedung Rektorat/Kampus-1 sekarang) berlantai 6 (enam) di jalan Dipati Ukur 114. Pembangunan Kampus baru ini dapat diselesaikan pada bulan Agustus 1999, sehingga pada awal perkuliahan bulan September 1999 telah dapat digunakan.

Mencermati dinamika peserta didik dan pengembangan Institusi kedepan, pada tanggal 24 Desember 1998 dibentuklah Yayasan Science dan Teknologi dan dilanjutkan dengan pengajuan pendirian STIMIK IGI dan STIE IGI ke DIKTI.

Pada bulan Juli 1999 STIE IGI diresmikan dengan keluarnya SK Mendiknas no. 119/D/O/1999 dengan 5 program studi : Akuntansi S1, Manajemen S1, Manajemen Pemasaran D3, Keuangan Perbankan D3 serta Akuntansi D3.

Pada bulan Agustus 1999 STIMIK IGI diresmikan dengan keluarnya SK Mendiknas no. 143/D/O/1999 dengan 5 program studi : Teknik Informatika S1,

Manajemen Informatika D3, Teknik Komputer D3, Komputerisasi Akuntansi D3 serta Teknik Informatika D3.

Agar Sistem Pendidikan lebih Efisien, Efektif, Produktif dengan Struktur Organisasi yang lebih baik, enam bulan kemudian dilakukan usulan ke DIKTI untuk melakukan Merger kedua Sekolah Tinggi diatas menjadi Universitas.

Pada hari Selasa, tgl. 8 Agustus 2000 keluarlah SK MENDIKNAS no. 126/D/O/2000 atas Universitas Komputer Indonesia yang disingkat dengan nama UNIKOM.Pada SK tersebut sekaligus diijinkan dibukanya 11 program studi baru: Teknik Komputer S1, Manajemen Informatika S1, Teknik Industri S1, Teknik Arsitektur S1, Perencanaan Wilayah dan Kota S1, Ilmu Hukum S1, Ilmu Komunikasi S1, Ilmu Pemerintahan S1, Desain Interior D3, Desain Komunikasi Visual S1 dan Desain Komunikasi Visual D3.

Sejak berdirinya pada tahun 2000, setiap tahunnya UNIKOM menerima ± 2.000 mahasiswa baru. Terakhir pada tahun 2009 yang lalu diterima sebanyak 3.108 mahasiswa baru. Hingga tahun akademik 2009/2010 terdapat 6 Fakultas dan 23 Program Studi di UNIKOM dengan jumlah mahasiswa sebanyak 15.000 orang yang berasal dari berbagai pelosok tanah air dan dari luar negeri yang sedang menempuh pendidikan di UNIKOM.

2.1.2 Visi, Misi, dan Tujuan Unikom

Berikut penjabaran dari visi, misi, dan tujuan Universitas Komputer Indonesia adalah:

1. Visi

Menjadi Universitas terdepan dibidang Teknologi Informasi & Komputer, berwawasan Global dan menjadi pusat Unggulan dibidang ilmu pengetahuan Teknologi dan seni yang mendukung pembangunan nasional serta berorientasi pada kepentingan masyarakat, bangsa dan Negara .

2. Misi

Menyelenggarakan Pendidikan tinggi kearah masyarakat Industri maju dengan sistem pendidikan yang kondusif, tenaga pengajar berkualitas dan program-program studi berbasis pada teknologi informasi & komputer dengan mengoptimalkan sumber daya yang ada, kualitas dan manajemen mutu berdasarkan prinsip *Quality Is Our Tradition*.

3. Tujuan

Menghasilkan Ilmuwan dan berpikiran tinggi maju dibidangnya masing-masing, mahir menggunakan teknologi informasi & komputer dalam bekerja serta beriman dan bertakwa kepada Tuhan Yang Maha Esa.

2.1.3 Logo Unikom

Logo Unikom (Gambar II.1) memiliki makna dari setiap bagiannya, yaitu bingkai segi lima, lingkaran dalam segi lima tempat tulisan berwarna kuning, bulatan dalam berwarna biru, komputer, stasiun relay, satelit, cakrawala, dan buku.



Gambar 2.1 Logo Unikom

Adapun penjabaran makna dari setiap bagiannya adalah sebagai berikut:

1. **Bingkai Segi Lima**

   Melambangkan UNIKOM berlandaskan falsafah negara yakni Pancasila dan Undang-Undang Dasar 1945.

2. **Lingkaran dalam Segi Lima Tempat Tulisan Berwarna Kuning**

   Melambangkan motto UNIKOM menuju kejayaan yakni *Quality Is Our Tradition.*

3. **Bulatan dalam Berwarna Biru**

   Melambangkan UNIKOM bertujuan menghasilkan ilmuwan unggul dan berpikiran maju yang Bertaqwa kepada Tuhan Yang Maha Esa.

4. **Komputer**

   Melambangkan ciri utama UNIKOM yang memberikan pendidikan Teknologi Informasi dan Komputasi pada seluruh Jurusan yang ada dilingkungan Universitas Komputer Indonesia, menjadi Universitas Terdepan dibidang Teknologi Informasi dan Komputer serta sebagai Universitas komputer pertama di Indonesia.

5. **Stasiun Relay**

   Melambangkan UNIKOM menyelenggarakan Pendidikan Tinggi kearah masyarakat industri maju dengan sistem pendidikan yang kondusif dan tenaga pengajar berkualitas untuk menghasilkan lulusan-lulusan terbaik.

6. **Satelit**

   Melambangkan UNIKOM berwawasan Global dan menjadi pusat unggulan dibidang IPTEK & seni yang mendukung Pembangunan Nasional serta berorientasi pada kepentingan masyarakat, bangsa dan negara.

7. **Cakrawala**

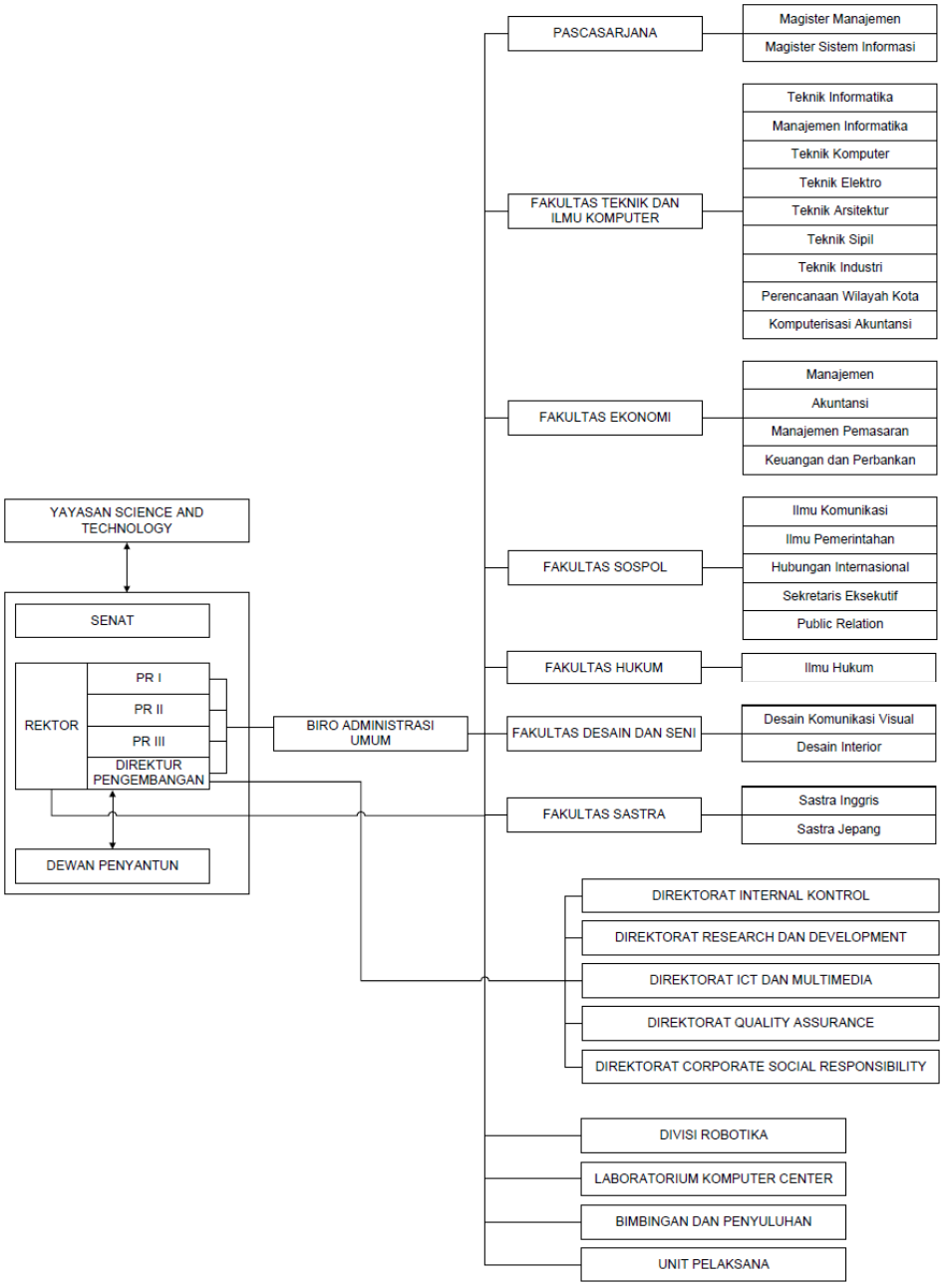
   Melambangkan indahnya menggapai Cita-cita dan mengejar ilmu setinggi Langit.

8. **Buku**

   Melambangkan sumber ilmu yang tiada habis-habisnya.

2.1.4 Struktur Organisasi Unikom

Unikom terdiri dari bagian atau divisi yang memiliki peranannya masing-masing.



Gambar 2.2 Struktur

Keterangan :

PR I : Pembantu Rektor Bidang Akademik

PR II : Pembantu Rektor Bidang Keuangan, Administrasi, dan Kepegawaian

PR III : Pembantu Rektor Bidang Kemahasiswaan dan Kerjasama

2.2 Landasan Teori

2.2.1 Absensi

Absensi merupakan proses pencatatan kehadiran peserta didik yang mengikuti kegiatan pembelajaran yang diselenggarakan serta dijadikan salah satu parameter penilaian prestasi peserta didik

2.2.2 Radio Frequency Identification (RFID)

1. Pengertian

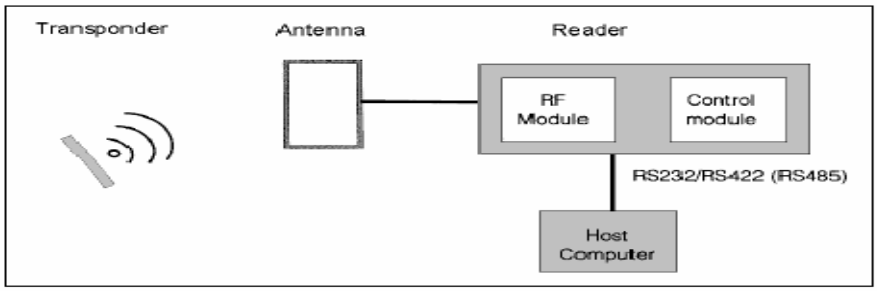
RFID atau Radio Frequency Identification, adalah suatu metode yang mana bisa digunakan untuk menyimpan atau menerima data secara jarak jauh dengan enggunakan suatu piranti yang bernama RFID tag atau transponder. Suatu RFID tag adalah sebuah benda kecil, misalnya berupastiker adesif, dan dapat itempelkan pada suatu barang atau produk. RFID tag berisi antena yang memungkinkan mereka untuk menerima dan merespon terhadap suatu query yang dipancarkan oleh suatu RFID transceiver

2. Sistem RFID

Pada sistem RFID umumnya, tag atau *transponder* ditempelkan pada suatu objek.Setiap tag membawa dapat membawa informasi yang unik, di antaranya: serial number, model, warna, tempat perakitan, dan data lain dari objek tersebut. Ketika tag ini melalui medan yang dihasilkan oleh pembaca RFID yang kompatibel, tag akan mentransmisikan informasi yang ada pada tag kepada pembaca RFID, sehingga proses identifikasi objek dapat dilakukan.

Sistem RFID terdiri dari empat komponen, di antaranya seperti dapat dilihat pada gambar 2.3:

1. Tag: Merupakan device yang menyimpan informasi untuk identifikasiobjek. Tag RFID sering juga disebut sebagai *transponder*.
2. Antena: untuk mentransmisikan sinyal frekuensi radio antara pembaca RFID dengan tag RFID.
3. Pembaca RFID: adalah devais yang kompatibel dengan tag RFID yang akan berkomunikasi secara *wireless* dengan tag.
4. Software Aplikasi: adalah aplikasi pada sebuah workstation atau PC yang dapat membaca data dari tag melalui pembaca RFID. Baik tag dan pembaca RFID diperlengkapi dengan antena sehingga dapat menerima dan memancarkan gelombang elektromagnetik.



Gambar 2.3 Sistem RFID

3. Pembaca RFID

Pembaca RFID adalah merupakan penghubung antara software aplikasi denganantena yang akan meradiasikan gelombang radio ke tag RFID. Gelombang radioyang diemisikan oleh antena berpropagasi pada ruangan di sekitarnya. Akibatnyadata dapat berpindah secara *wireless* ke tag RFID yang berada berdekatan denganantena.

Sebuah pembaca RFID harus menyelesaikan dua buah tugas, yaitu:

1. Menerima perintah dari software aplikasi
2. Berkomunikasi dengan tag RFID

4. Tag RFID

Tag RFID adalah devais yang dibuat dari rangkaian elektronika dan antena yangterintegrasi di dalam rangkaian tersebut. Rangkaian elektronik dari tag RFID umumnya memiliki memori sehingga tag ini mempunyai kemampuan untukmenyimpan data. Memori pada tag secara dibagi menjadi sel-sel. Beberapa selmenyimpan data *Read Only*, misalnya *serial number* yang unik yang disimpanpada saat tag tersebut diproduksi. Sel lain pada RFID mungkin juga dapat ditulis

dan dibaca secara berulang.

Berdasarkan catu daya tag, tag RFID dapat digolongkan menjadi:

1. Tag Aktif: yaitu tag yang catu dayanya diperoleh dari batere, sehingga akan mengurangi daya yang diperlukan oleh pembaca RFID dan tag dapat mengirimkan informasi dalam jarak yang lebih jauh. Kelemahan dari tipe tag ini adalah harganya yang mahal dan ukurannya yang lebih besar karena lebih komplek.

2.2.3. Sistem

2.2.3.1 Pengertian Sistem

Pada dasarnya kata sistem berasal dari bahasa Yunani “Sistema” yang berarti kesatuan yaitu kesuluruhan dari bagian-bagian yang mempunyai hubungan satu sama lain.Sistem adalah suatu kumpulan dari elemen-elemen baik berbentuk fisik maupun bukan fisik yang menunjukkan suatu hubungan diantaranya dan berinteraksi bersama-sama menuju suatu tujuan.Suatu sistem dapat terdiri dari beberapa subsistem yang saling berhubungan membentuk suatu kesatuan sehingga tujuan dari sistem dapat tercapai.

Berikut beberapa pengertian sistem menurut para ahli yaitu :

a. Stephen A.Moscove dan Mark G.Simkin

   Suatu sistem adalah suatu kesatuan yang terdiri dari interaksi sub sistem yang berusaha untuk mencapai tujuan yang sama.

b. MJ.Alexander

   Suatu sistem adalah suatu group dari elemen-elemen baik berbentuk fisik maupun yang menunjukkan suatu kumpulan saling berhubungan

diantaranya dan berinteraksi bersama-sama menuju satu atau lebih tujuan akhir dari sistem.

c. Richard F.Newschell

Suatu sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan dan dikembangkan sesuai dengan skema yang berinteraksi untuk melaksanakan suatu kegiatan utama dalam bisnis.

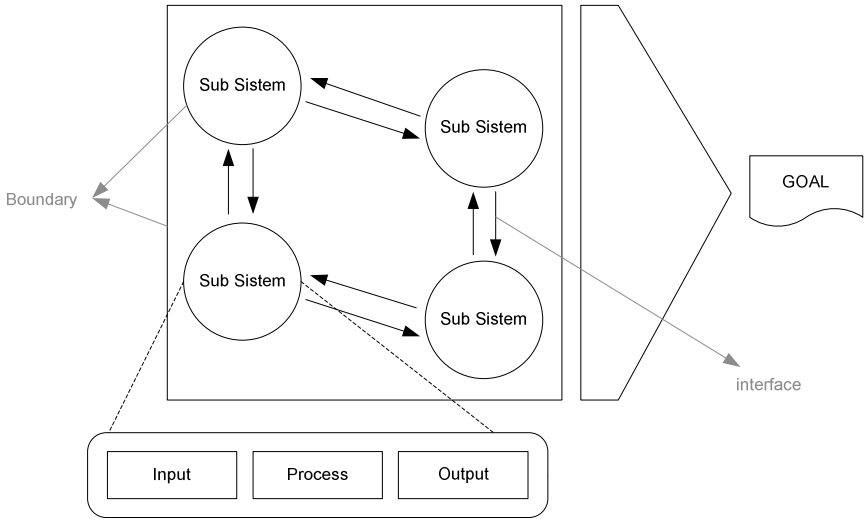
Ciri – ciri dari sistem yaitu :

a. Adanya tujuan yang jelas.
b. Mempunyai strukutur yang jelas.
c. Terdiri dari satu kesatuan usaha dari bagian-bagian yang saling bergantung dan berinteraksi satu sama lain.

2.2.3.2 Karakteristik Sistem

Karakteristik sistem adalah sistem yang mempunyai komponen sistem,batasan sistem,linkungan luar sistem,penghubung,input,output,pengolah sistem dan sasaran sistem.

Untuk lebih jelasnya dapat dilihat dari gambar dibawah ini.



Gambar 2.4 Karakterisitik Sistem

Berikut penjelasan dari gambar diatas yaitu :

1. Komponen Sistem (*Components*)

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi yaitu saling bekerjasama membentuk satu kesatuan.Suatu sistem dapat mempunyai sistem yang lebih besar yang disebut Supra Sistem.

2. Batasan Sistem (*Boundary*)

   Batasan sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau lingkungan luarnya.Dimana suatu sistem dipandang sebagai satu kesatuan yang menunjukkan ruang lingkup dari sistem tersebut.

3. Lingkungan Luar Sistem (*Environment*)

   Lingkungan luar sistem adalah lingkungan diluar batas dari sistem yang mempengaruhi operasi sistem yang dapat bersifat menguntungkan atau merugikan.

4. Penghubung Sistem (*Interface*)

   Penghubung sistem merupakan penghubung antara satu subsistem dengan subsistem lainnya yang memungkinkan adannya sumber daya mengalir dari satu subsistem dengan subsistem yang lainnya.

5. Masukkan Sistem (*Input*)

   Masukkan sistem adalah energy yang dimasukkan ke dalam sistem yang dapat berupa masukkan perawatan (*Maintenance*) dan masukkan sinyal (*Signal Input*).

6. Keluaran Sistem (*Ouput*)

   Keluaran sistem adalah hasil dari energy yang diolah dan diklasifikasikan menjadi keluaran yang berguna dari sisa pembuangan. Keluaaran dapar berupa masukan dari subsistem yang lain atau kepada supra sistem.

7. Pengolah Sistem (*Process*)

   Pengolah sistem akan mengubah masukan menjadi keluaran.

8. Sasaran Sistem (*Goal*)

   Sasaran sistem sangat menentukan sekali masukan yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem. Suatu sistem dikatakan berhasil jika mengenai sasaran atau tujuan.

2.2.4. Data dan Informasi

2.2.4.1 Pengertian Data

Data adalah deskripsi tentang benda, kejadian, aktivitas, dan transaksi yang tidak mempunyai makna atau tidak berpengaruh secara langusng kepada pemakai. Data dapat berupa nilai yang terformat, teks, citra, audio, dan video.

2.2.4.2 Pengertian Informasi

Informasi merupakan hasil proses data-data yang beragam yang telah dibentuk sedemikian rupa sehingga sesuai dengan keinginan pengguna dan telah mengalami proses yang telah tersusun dengan baik dan benar sesuai kriteria yang diharapkan.

Ciri-ciri informasi adalah sebagai berikut :

a. Data yang telah diolah.
b. Menjadi suatu bentuk yang lebih berguna dan berarti bagi yang menerima.
c. Menggambarkan suatu kejadian dan kesatuan nyata.
d. Digunakan untuk mengambil keputusan.

Ada beberapa definisi informasi diantaranya sebagai berikut :

a. Informasi adalah pernyataan-pernyataan atau bentuk-bentuk yang dapat digunakan untuk pengambilan keputusan.
b. Informasi adalah data yang telah diletakkan dalam konteks yang lebih berarti dan berguna yang dikomunikasikan kepada penerima untuk digunakan didalam pembuatan keputusan.

2.2.4.3 Kualitas Informasi

Istilah kualitas informasi terkadang juga dipakai untuk menyatakan informasi yang baik.Kualitas informasi dapat diukur dari tiga hal yaitu :

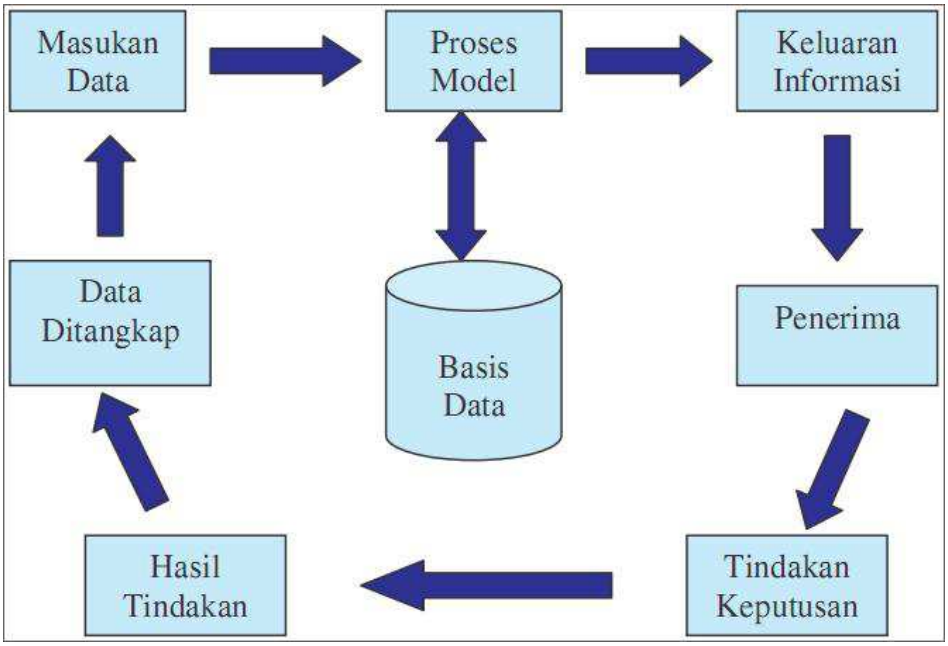
a. Akurat berarti suatu informasi harus bebas dari kesalahan-kesalahan dan tidak menyesatkan serta jelas dalam menjelaskan maksudnya sehingga hasil dari informasi dapat dipergunakan sebagai acuan bagi pengguna informasi untuk mengambil keputusan..
b. Tepat waktu berarti informasi yang diterima oleh penerima tidak boleh terlambat. Suatu informasi harus sesuai dengan keadaan saat itu. Keterlambatan suatu informasi dapat berakibat fatal bagi suatu organisasi atau penggunanya. Hal ini dikarenakan informasi merupakan landasan dalam pengambilan keputusan.
c. Relevan berarti informasi harus memiliki manfaat bagi penggunanya. Relevansi informasi untuk setiap orang berbeda-beda tergantung pada pengunannya.

2.2.4.4 Siklus Informasi

Siklus informasi menurut John Burch adalah gambaran secara umum mengenai proses terhadap data sehingga menjadi informasi yang bermanfaat bagi pengguna. Informasi yang menghasilkan informasi berikutnya. Demikian seterusnya proses pengolahan data menjadi informasi.

Data ditangkap sebagai *input*, diproses melalui suatu model membentuk *output* berupa informasi. Pemakai kemudian menerima informasi tersebut sebagai

landasan untuk membuat suatu keputusan dan melakukan tindakan operasional yang akan membuat sejumlah data baru. Data baru tersebut selanjutnya menjadi input pada proses berikutnya, begitu seterusnya sehingga membentuk suatu siklus



informasi/*Information Cycle.*

Gambar 2.5 Siklus Informasi

2.2.5. Sistem Informasi

Menurut John F. Nash (1995:8) yang diterjemahkan oleh LaMidjan dan Azhar Susanto, menyatakan bahwa Sistem Informasi adalah kombinasi dari manusia, fasilitas atau alatteknologi, media, prosedur dan pengendalian yang bermaksud menata jaringan komunikasi yang penting,proses atas transaksi-transaksi tertentu dan rutin, membantu manajemen dan pemakai intern dan ekstern dan menyediakan dasar pengambilan keputusan yang tepat.

Sedangkan menurut Henry Lucas (1988:35) yang diterjemahkan oleh Jugianto H.M, menyatakan bahwa sistem Informasi adalah suatu kegiatan dari prosedur prosedur yang diorganisasikan, bilamana dieksekusi akan menyediakan informasi untuk mendukung pengambilan keputusan dan pengendalian di dalam organisasi.

Menurut John F.Nash dan Martil B.Robert (1988:35) yang diterjemahkan oleh Jugianto H.M, menyatakan bahwa sistem Informasi adalah kombinasi dari orang-orang, fasilitas, teknologi, media, prosedur-prosedur dan pengendalian yang ditujukan untuk mendapatkan jalur komunikasi pentingm, memproses tipe transaksi rutin tertentu, memberi sinyal kepada manajemen yang lainnya terhadap kejadian-kejadian internal.

Dari ketiga pengertian sitem informasi diatas, maka dapat disimpulkan bahwa Sistem Informasi menyediakan informasi untuk membantu pengambilan keputusan manajemen, operasi perusahaan dari hari ke hari dan informasi yang layak untuk pihak luar perusahaan.

2.2.6. Rekayasa Perangkat Lunak

Rekayasa perangkat lunak (software engineering) merupakan pembangunan dengan menggunakan prinsip atau konsep rekayasa dengan tujuan menghasilkan perangkat lunak yang bernilai ekonomis yang dipercaya dan bekerja secara efisien menggunakan mesin. Perangkat lunak banyak dibuat dan pada akhirnya sering tidak digunakan karena tidak memenuhi kebutuhan pelanggan atau bahkan karena masalah non-teknis seperti keengganan pemakai

perangkat lunak (user) untuk mengubah cara kerja manual ke otomatis, atau ketidakmampuan user menggunakan komputer. Oleh karena itu, rekayasa perangkat lunak dibutuhkan agar perangkat lunak yang dibuat tidak hanya menjadi perangkat lunak yang tidak terpakai.

Dalam rekayasa perangkat lunak terdapat tiga fase, yakni :

1. **Fase pendefinisian**

   Fase pendefinisian yang fokus pada “what” yang artinya harus mencari tahu atau mengidentifikasi informasi apa saja yang harus diproses, seperti apa fungsi dari performansi yang diinginkan, seperti apa yang perilaku sistem yang diinginkan, apa kriteria validasi yang dibutuhkan untuk mendefinisikan sistem.

2. **Fase pengembangan**

   Fase pengembangan yang fokus pada “how” yang artinya selama tahap pengembangan perangkat lunak seorang perekayasa perangkat lunak (software engineer) berusaha untuk mendefinisikan bagaimana data distrukturkan dan bagaimana fungsi – fungsi yang dibutuhkan diimplementasikan di dalam arsitektur perangkat lunak, bagaimana detail prosedural diimplementasikan, bagaimana karakter antarmuka tampilan, bagaimana desain ditranslasikan ke dalam bahasa pemrograman dan bagaimana pengujian akan dijalankan.

3. **Fase pendukung (*support phase*)**

   Fase pendukung fokus pada perubahan yang terasosiasi pada perbaikan kesalahan (*error*), adaptasi yang dibutuhkan pada lingkungan yang

terlibat, dan perbaikan yang terjadi akibat perubahan kebutuhan pelanggan (*customer*). Fase pendukung terdiri dari empat tipe perubahan, antara lain :

a) Koreksi (*Correction*)

   Walaupun dengan jaminan kualitas yang terbaikm akan selalu ada kecacatan atau keinginan pelanggan (*customer*) yang tidak tertangani oleh perangkat lunak. Pemeliharaan dengan melakukan perbaikan terhadap kecacatan perangkat lunak.

b) Adaptasi (*adaption*)

   Pada saat tertentu lingkungan asli (seperti CPU,sistem operasi, aturan bisnis, karakteristik produk luar) dimana perangkat lunak dikembangkan akan mengalami perubahan. Pemeliharaan adaptasi merupakan tahap untuk memodifikasi perangkat lunak guna mengakomodasi perubahan lingkungan luar dimana perangkat lunak dijalankan.

c) Perbaikan (*enhancement*)

   Sejalan dengan digunakannya perangkat lunak, maka pelanggan (*customer*) atau pemakainya (*user*) akan mengenali fungsi tambahan yang dapat mendatangkan manfaat. Pemeliharaan perfektif atau penyempurnaan melakukan ekstensi atau penambahan pada kebutuhan fungsional sebelumnya.

d) Pencegahan (*prevention*)

   Keadaan perangkat lunak komputer sangat dimungkinkan untuk perubahan. Oleh karena itu, pemeliharaan pencegahan *(preventive)*atau sering disebut juga dengan rekayasa ulang sistem (*software*

reengineering) harus dikondisikan untuk mampu melayani kebutuhan pemakainya. Untuk menanggulangi hal ini maka perangkat lunak harus dirancang dan dikondisikan untuk mengakomodasi perubahan kebutuhan yang diinginkan oleh pemakainya. Di lain sisi biasanya setelah perangkat lunak dikirimkan ke *user*maka masih dibutuhkan asistensi dan *help desk* dari pengembang perangkat lunak.

2.2.6.1 Software development life cycle (SDLC)

Software development life cycle (SDLC) atau sering disebut dengan *Sistem development life cycle*adalah proses mengembangkan atau mengubah suatu sistem perangkat lunak dengan menggunakan model – model dan metodologi yang digunakan orang untuk mengembangkan sistem – sistem perangkat lunak sebelumnya.

Tahapan – tahapan yang ada pada SDLC secara global adalah sebagai berikut :

1. Inisiasi (*initiation)*

   Tahap ini biasanya ditandi dengan pembuatan proposal proyek perangkat lunak.

2. Pengembangan konsep sistem (*sistem concept developtment)*

   Mendefinisikan lingkup konsep termasuk dokumen lingkup sistem, analisis manfaat biaya, manajemen rencana, dan pembelajaran kemudahan sistem.

3. Perencanaan (*Planning)*

   Mengembangkan rencana manajemen proyek dan dokumen perencanaan lainnya. Menyediakan dasar untuk mendapatkan sumber daya (*Resource)* yang dibutuhkan untuk memperoleh solusi.

4. Analisis kebutuhan (*requirements analysis)*

   Menganalisis kebutuhan pemakai sistem perangkat lunak (*User)* dan mengembangkan kebutuhan *user*. Membuat dokumen kebutuhan fungsional.

5. Desain (*Design)*

   Mentransformasikan kebutuhan detail menjadi kebutuhan yang telah lengkap, dokumen desain sistem fokus pada bagaimana yang dapat memenuhi fungsi – fungsi yang dibutuhkan.

6. Pengembangan (*Development)*

   Mengkonversi desain ke sistem informasi yang lengkap termasuk bagaimana memperoleh dan melakukan instalasi lingkungan sistem yang dibutuhkan, membuat basis data dan mempersiapkan berkas atau *file*pengujian, pengodean, pengompilasian, memperbaiki dan membersihkan program, peninjauan pengujian.

7. Integrasi dan pengujian (*integration and test)*

   Mendemonstrasikan sistem perangkat lunak bahwa telah memenuhi kebutuhan yang dispesifikasikan pada dokumen kebutuhan fungsional. Dengan diarahkan oleh staf penjamin kualitas dan *user*. Menghasilkan laporan analisis pengujian.

8. Implementasi (*implementation)*

   Termasuk pada persiapan implementasi, implementasi perangkat lunak pada lingkungan produksi dan menjalankan resolusi dari permasalahan yang teridentifikasi dari fase integrasi dan pengujian.

9. Operasi dan pemeliharaan (*operation and maintenance)*

   Mendeskripsikan pekerjaan untuk mengoperasikan dan memelihara sistem informasi pada lingkungan produksi (lingkungan pada user) termasuk implementasi akhir dan masuk pada proses peninjauan.

10. Disposisi (*Disposition)*

    Mendeskripsikan aktifitas akhir dari pengembangan sistem dan membangun data yang sebenarnya sesuai dengan aktifitas *user*.

2.2.7. Perancangan Berorientasi Objek

Pendekatan berorientasi objek merupakan suatu teknik atau cara pendekatan dalam melihat permasalahan dan sistem (sistem perangkat lunak, sistem informasi atau sistem lainya). Pendekatan berorientasi objek akan memandang sistem yang akan dikembangkan sebagai suatu kumpulan objek yang berkorespondensi dengan objek – objek dunia nyata.

Ada banyak cara untuk mengabstraksikan dan memodelkan objek – objek tersebut, mulai dari abstraksi objek, kelas, hubungan antar kelas sampai abstraksi sistem. Pada saat mengabstraksikan dan memodelkan objek, data, dan proses – proses yang dipunyai oleh objek akan dienkapsulasi (dibungkus) menjadi satu kesatuan.

Dalam rekayasa perangkat lunak, konsep pendekatan berorientasi objek dapat diterapkan pada tahap analisis, perancangan, pemrograman, dan pengujian perangkat lunak. Ada berbagai teknik yang dapat digunakan pada masing - masing tahap tersebut, dengan aturan dan alat bantu pemodelan tertentu.

Sistem berorientasi objek merupakan sebuat sistem yang dibangun dengan berdasarkan metode berorientasi objek adalah sebuah sistem yang komponennya dibungkus menjadi kelompok data dan fungsi. Setiap komponen dalam sistem tersebut dapat mewarisi atribut dan sifat dan komponen lainya, serta dapat berinteraksi satu sama lain.

Berikut adalah komponen – komponen yang terdapat pada sistem berorientasi objek :

1. Kelas (*Class)*

   Kelas adalah kumpulan objek – objek dengan karakteristik yang sama. Kelas merupakan definisi statis dan himpunan objek yang sama yang mungkin lahir atau diciptakan dari kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (metode), hubungan (*relationship)* dan arti. Suatu kelas dapat diturunkan dan kelas yang lain, dimana atribut dan kelas semula dapat diwariskan ke kelas yang baru.

   Secara teknis, kelas adalah sebuah struktur tertentu dalam pembuatan perangkat lunak.Kelas merupakan bentuk terstruktur pada kode program yang menggunakan metodologi berorientasi objek.

2. Objek (*object)*

   Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, status, atau hal – hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampy menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan dimanipulasi dan dihancurkan.

   Secara teknis, sebuah kelas saat program dieksekusi maka akan dibuat sebuah objek. Objek dari segi teknis adalah elemen pada saat *runtime* yang akan diciptakan, dimanipulasi, dan dihancurkan saat dieksekusi sehingga sebuah objek hanya pada saat sebuah program dieksekusi, jika masih dalam bentuk kode, disebut sebagai kelas jadi pada saat *runtime* (saat sebuah program dieksekusi) yang kita punya adalah objek, di dalam teks program yang kita lihat hanyalah kelas ilustrasi kelas dan objek.

3. Metode (*Method)*

   Operasi atau metode pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode yang berfungsi untuk memanipulasi objek itu sendiri. Metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek.

4. Atribut (*Attribute)*

   Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen – elemen data yang dimiliki oleh

objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek.

5. Abstraksi (*Abstraction)*

   Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek – aspek lain yang tidak sesuai dengan permasalahan.

6. Enkapsulasi (*Encapsulation)*

   Pembungkusan atribut data dan layanan (operasi – operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya,

7. Pewarisan (*Inheritance)*

   Mekanisme yang memungkinkan saatu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dirinya.

8. Antarmuka (*interface)*

   Antarmuka atau *interface* sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah *interface*dapat diimplementasikan oleh kelas lain.

   Sebuah kelas dapat mengimplementasikan lebih dari satu antarmuka dimana kelas ini akan mendeklarasikan metode pada antarmuka yang dibutuhkan oleh kelas itu sekaligus mendefinisikan isinya pada kode program kelas itu. Metode pada antarmuka yang diimplementasikan pada suatu kelas harus sama persis dengan yang ada pada antarmuka. *Interface* biasanya digunakan

agar kelas yang lain tidak mengakses langsung ke suatu kelas mengakses *interface.*

9. *Reusability*

   Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

10. Generalisasi dan Spesialisasi

    Menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus. Misalnya kelas yang lebih umum (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil,motor dan kereta.

11. Komunikasi antarobjek

    Komunikasi antar objek dilakukan lewat pesan (*Message*) yang dikirim dan satu objek ke objek lainnya.

12. Polimorfisme (*Polymorfism)*

    Kemampuan suatu objek untuk digunakan di banyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program

13. *Package*

    Package adalah sebuah kontainer atau kemasan yang dapat digunakan untuk mengelompokkan kelas - kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam *package* yang berbeda.

2.2.8. Basis Data

Basis data terdiri atas dua kata yaitu basis dan data.Basis kurang lebih dapat diartikan sebagai markas atau gudang, tempat bersarang atau berkumpul. Sedangkan data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai,peserta,pembeli,pelanggan), barang, hewan, peristiwa konsep, keadaan dan sebagainya, yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya.

Basis data(*database*) sendiri dapat didefiniskan sebagai berikut: Basis data (*database*) merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan pada perangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya.

Prinsip utama pada basis data adalah pengaturan data/arsip dan tujuan utamanaya adalah kecepatan dan kemudahan dalam pengambilan kembali data/arsip yang menggunakan media penyimpanan elektronis seperti disk (disket atau harddisk).Basis data dikelola/ditangani melalui perantara alat/mesin pintar elektronis yang kita kenal sebagaikomputer.Basis data bukan hanya sekedar penyimpanan data secara elektronis dengan bantuan komputer. Artinya, tidak semua bentuk penyimpanan data secara elektronis bisa disebut basis data.Hal sangat ditonjolkan dalam basis data adalah pengaturan atau penilaian atau pengelompokan atau pengorganisasian data yang akankita simpan sesuai fungsi atau jenisnya.

Penilaianatau pengelompokkan atau pengorganisasian ini dapat berbentuk sejumlah*file* atau*tabel* terpisah atau dalam bentuk pendefinisian kolom-kolom atau

field-field data dalam setiap *file* atau *tabel.* Operasi-operasi dasar yang dapat dilakukan berkenaan dengan basis data dapat meliputi pembuatan basis data baru (*create database*), penghapusan basis data(*drop database*), pembuatan file atau tabel dari suatu basis data(*create tabel*), penghapusan file atau tabel dari suatu basis data(*drop tabel*), pengisian/penambahan data baru di sebuah basis data(*insert*), pengambilan data dari sebuah*file* atau *tabel*(*retrieve/search),* pengubahan data dari sebuah*file* atau *tabel*(*update),* dan penghapusan data dari sebuah *file* atau *tabel*(*delete*).

2.2.8.1. Database Management Sistem (DBMS)

Pengelolaan basis data secara fisik tidak dilakukan oleh pemakai secara langsung, tetapi ditangani oleh sebuah Perangkat Lunak (Sistem) yang khusus/spesifik. Perangkat lunak inilah (disebut DBMS) yang akan menentukan bagaimana data diorganisasi, disimpan, diubah dan diambil kembali. Ia juga menerapkan mekanisme pengamanan data, pemakaian data secara bersama, pemaksaan keakuratan/konsistensi data, dan sebagainya.

Perangkat lunak yang termasuk DBMS seperti dBase III+, dBase IV, FoxBase, Rbase, MS-Access dan Borland-Paradox (untuk kelas sederhana) atau Borland-Interbase, MS-SQLServer, CA-Open Ingres, Oracle, Informix dan Sybase (untuk kelas kompleks/berat).

2.2.8.2. Tujuan Basis Data

Tujuan awal dan utama dalam pengelolaan data dalam sebuah basis data adalah agar dapat memperoleh menemukan kembali data (yang dicari) dengan mudah dan cepat. Di samping itu, pemanfaatan basis data untuk pengelolaan data, juga memiliki tujuan-tujuan lain.

Secara lebih lengkap, pemanfaatan basis data dilakukan untuk memenuhi sejumlah tujuan (objektif) seperti berikut ini:

a. Kecepatan dan kemudahan (speed)

b. Efisiensi ruang penyimpanan (space)

c. Keakuratan (accuracy)

d. Ketersediaan (availability)

e. Kelengkapan (completeness)

f. Keamanan (security)

g. Kebersamaan pemakaian (sharability)

2.2.8.3. Pemodelan Data

1. ERD (*Entity Relationship Diagram)*

Entity Relationship Diagram atau biasa dikenal dengan diagram E-R secara grafis menggambarkan isi sebuah *database*. Diagram inimemiliki duakomponen utama yaitu *entity* dan relasi. Untuk melambangkan fungsi diatas maka digunakan simbol-simbol yang bisa dilihat pada daftar simbol.

Diagram hubungan entitas atau yang lebih dikenal dengan sebutan E-R diagram, adalah notasi grafik dari sebuah model data atau sebuah model jaringan

yang menjelaskan tentang data yang tersimpan (*storage data*) dalam sistem secara abstrak.Diagram hubungan entitas tidak menyatakan bagaimana memanfaatkan data, membuat data, mengubah data dan menghapus data.

Terdapat tiga macam kardinalitas relasi, yaitu :

1. Relasi satu-ke-satu (*one-to-one*)
2. Relasi satu-ke-banyak (*one-to many)* atau banyak-ke-satu (*many-to-one)*
3. Relasi banyak-ke-banyak (*many-to-many)*

2. Kamus Data

Salah satu komponen kunci dalam sistem manajemen *database* adalah *file* khusus yang disebut kamus data (*data dictionary*). Kamus data didefinisikan sebagai berikut :

"Katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi".[*http://educozone.com/se/waterfall_model.html*].

Dengan menggunakan kamus data, analisis sistem dapat mendefinisikan data yang mengalir berisi informasi tentang struktur*database*. Untuk setiap elemen data yang disimpan dalam *database* seperti nomor pokok pegawai, diuraikan secara lengkap mulai dari nama, tempat penyimpanan, program komputer yang berhubungan dan lain-lain.

2.2.8.4. Data Definition Language (DDL)

Pada DDL ha terpenting adalah pembentukan database dengan objek utama tabel, indeks, mengubah tabel, menghapus tabel dan membuat domain.

Domain dalam hal ini adalah kemampuan database untuk membentuk suatu tipe baru yang pada dasarnya mengacu pada tipe dasar.

2.2.8.5. Data Manipulation Language (DML)

Pada DML hal terpenting adalah melakukan operasi atau manipulasi terhadap database dengan objek utama , *delete, selection,* pembentukan *view* dan *updt.* pembentukan *view* sebenarnya tidak dilakukan didalam served DDL karena *view* bukan merupakan *base tabels*. Dalam hal ini tak ada tabel yang secara fisik terbentuk dengan digunakan *view*.

2.2.8.6. Data Control Language (DCL)

Pada DCL hal terpenting adalah member otoritas, konfirmasi penyimpanan *database,* menunda perubahan dan menghapus otoritas yang telah diberikan. Pada beberapa database juga dapat diatur *database space* per user

2.2.9. Jaringan komputer

2.2.9.1 Pengertian Jaringan Komputer

Jaringan computer adalah kumpulan sejumlah pheripheral yang terdiri dari beberapa komputer , printer, LAN Card, dan peralatan lain yang saling terintegrasi satu sama lain. Dengan demikian dapat melakukan aktivitas sperti tukar-menukar data atau informasi dengan mudah dan dalam waktu yang singkat.

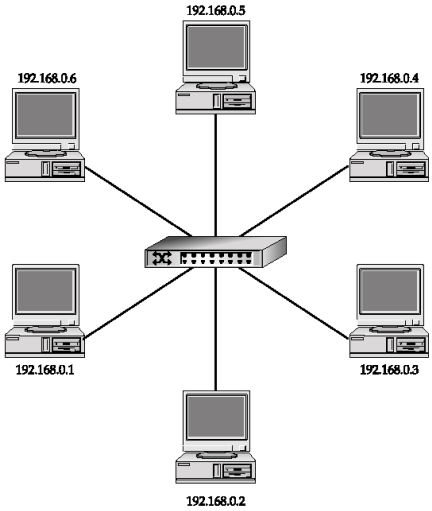
2.2.9.2 Jenis-jenis Jaringan Komputer

Macam jaringan komputer bila dilihat berdasarkan lingkup dan luas jangkauannya, dibedakan menjadi beberapa macam:

1. Local Area Network(LAN)

Merupakan suatu jaringan komputer yang masih berada didalam gedung atau ruangan. Dalam membuat jaringan LAN, minimal kita harus menyediakan dua buah komputer yang masing-masing memiliki kartu jaringan atau Lan Card.

Beberapa keuntungan menggunakan LAN adalah:



Gambar 2.8 Skema LAN

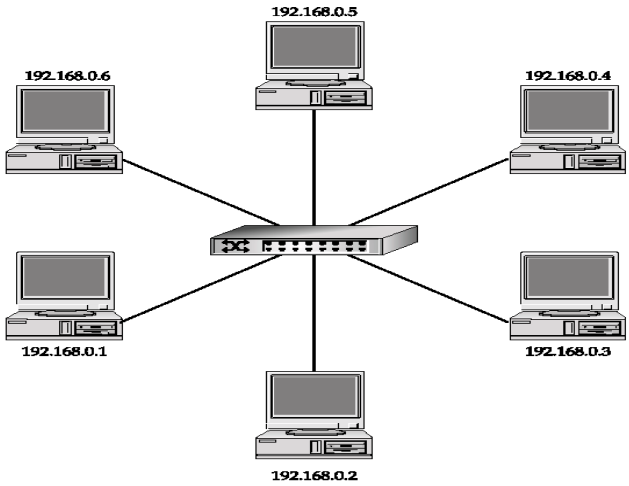
a. Dapat menghubungkan komputer dalam jumlah banyak
b. Akses antar komputer, baik untuk tukar-menukar data atau yang lain, berlangsung cepat dan mudah.
c. Dapat saling bertukar informasi dengan pengguna di luar area apabila terhubung dengan internet.

d. Dapat membackup data pada komputer lain tanpa harus membongkar hardisk

e. Hemat waktu dan biaya pengiriman paket data

2. Metropolitan Area Netwotk (MAN)

Terdiri dari beberapa jaringan LAN yang saling berhubungan. Letak jaringan ini bisa saling berjauhan tergantung dari panjangnya kabel yang kita gunakan. Jaringan ini juga dapat menjangkau lokasi yang berbeda tempat. MAN biasanya digunakan oleh sebuah perusahaan jaringan komputer dalam satu kota, antar kampus atau universitas, dan lain-lain

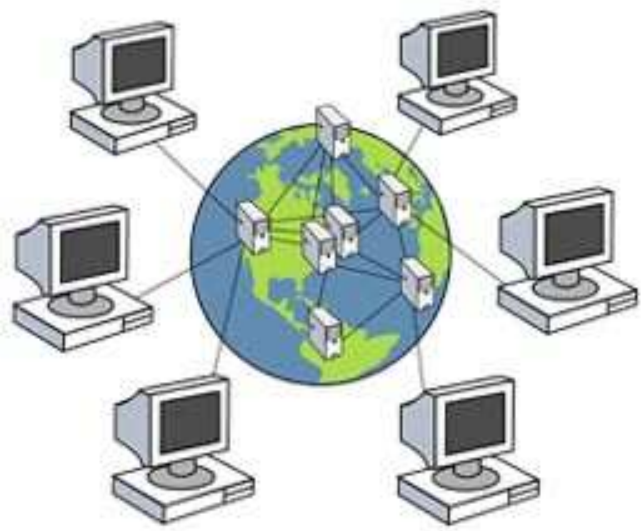


Gambar 2.9 Skema LAN

3. Wide Area Network (WAN)

Wide Area Network (WAN) merupakan bentuk jaringan komputer yang terdiri dari LAN dan Man. Jaringan WAN telah memenuhi berbagai kebutuhan sistem jaringan, seperti jaringan untuk publik, jaringan pada bidang

perbankan, jaringan jual-beli secara online di intenet, jaringan penjualan jasa, dan jaringan lainnya. Wan menggunakan protocol internet berupa *Network Service Provider* (NSP). Tanpa NSP, maka jaringan WAN tidak akan dapat bekerja. Dengan adanya NSP yang dihubungkan dengan jaringan WAN, maka akan membentuk suatu jaringan internet yang bersifat global. Dengan demikian, internet dapat diakses oleh orang yang akan memakai jaringan tersebut.



Gambar 2.10 SkemaWAN

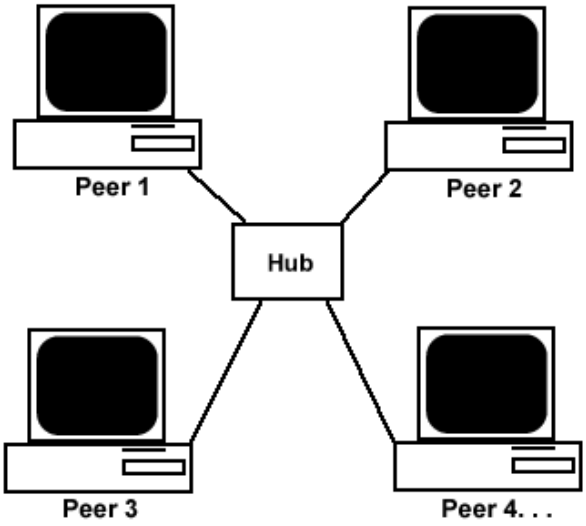
Jaringan WAN hanya menekankan pada fasilitas kecepatan akses transmisi sehingga memungkiknkan seluruh komunikasi dapat berjalan secara lancer serta efisien. Disamping itu WAN berfungsi untuk mengontrol jumlah lalu-lintas data dan mencegah penundaan (delay) yang berlebihan, sehingga transfer data lebih cepat.

2.2.10 Model/Type Jaringan

Berikut beberapa model jaringan yang didasarkan pada metode akses dan pemrosesan datanya :

1. Model Jaringan Peer TO Peer

Pada tipe jaringan ini pertukaran data hanya dapat kita lakukan antar dua komputer dalam satu area saja. Jaringan ini bias kita buat dengan menghubungkan dua komputer melalui kabel jaringan tipe *crossover* (khusus untuk dua komputer), atau menggunakan kabel *straight* yang terhubung dengan hub atau switch (untuk komputer satu area).



Gambar 2.14 Skema jaringan peer to peer

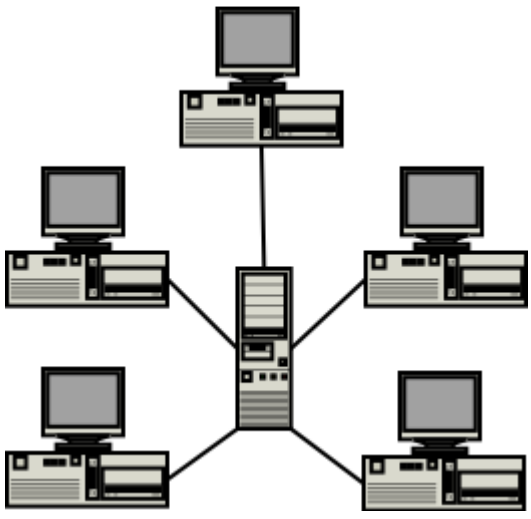
Dalam hirarki jaringan, komputer yang terhubung dalam tipe jaringan ini memiliki kedudukan yang sama anatara satu dengan yang lainnya. Artinya, setiap komputer didalam jaringan ini dapat bertindak sebagai server

maupun sebagai client. Jadi jaringan tidak terpaku pada satu komputer saja yang bertindak sebagai server.

2. Model jaringan Client/Server

Tipe jaringan jaringan ini terdiri dari sejumlah komputer dengan memakai satu atau beberapa komputer yang dijadikan server dan dihubungkan dengan sejumlah client. Jadi pada jaringan ini bias terdapat satu atau lebih dari satu server untuk mengendalikan beberapa komputer client.

Pada jaringan ini komputer server hanya bertugas memberikan service-service seperi database server, file server, dan lain sebagainya. Sementara komputer client pada jaringan ini hanya berhubungan dengan data pribadi server tidak bias diakses oleh client kecuali client tersebut mendapatkan hak akses.



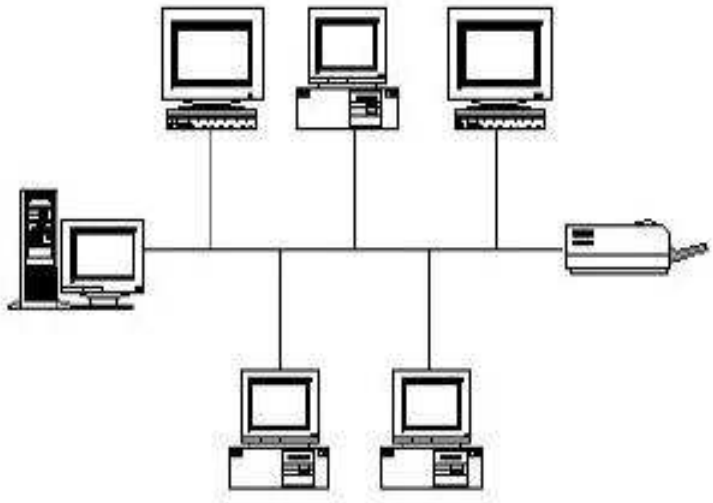
Gambar 2.15 Skema jaringan client-server

2.2.11 Arsitektur Jaringan Komputer

Arsitektur fisik jaringan identik dengan topologi yang akan digunakan dalam jaringan tersebut. Hal tersebut bertujuan agar apabila suatu saat jaringan tersebut ingin kita kembangkan menjadi suatu jaringan dengan skala yang lebih besar dan luas maka pemasangan maupun perawatan jaringan menjadi lebih mudah.

1. Topologi Bus

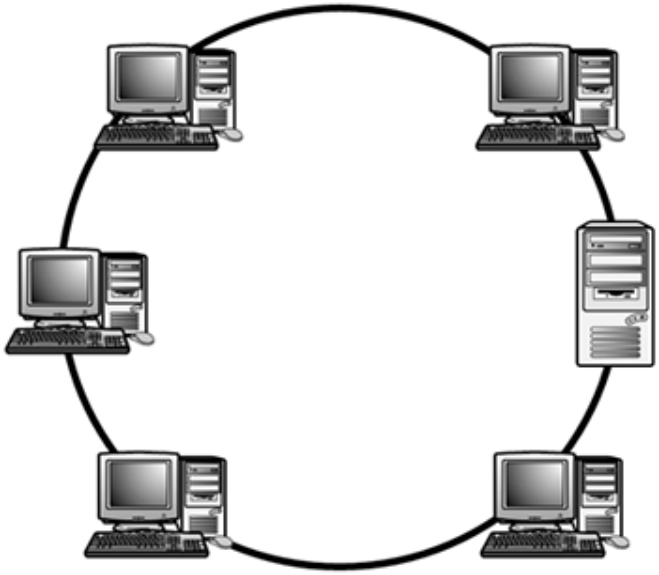
Pada tipe jaringan topologi bus, masing-masing server dan workstation dihubungkan pada sebuah kabel yang disebut bus. Kabel untuk menghubungkan jaringan ini biasanya menggunakan kabel coaxcial (contoh kabel BNC). Setiap server dan workstation yang tersambungkan pada bus menggukan konekor T.



Gambar 2.16 Topologi Bus

2. Topologi Token Ring (Ring)

Pada tipe jaringan topologi ring semua jaringan yang terhubung dari satu komputer lain berkeliling membentuk suatu lingkaran (*loop*). Pada toplogi ini komputer server dapat kita letakan dimana saja sesuai dengan keinginan kita.

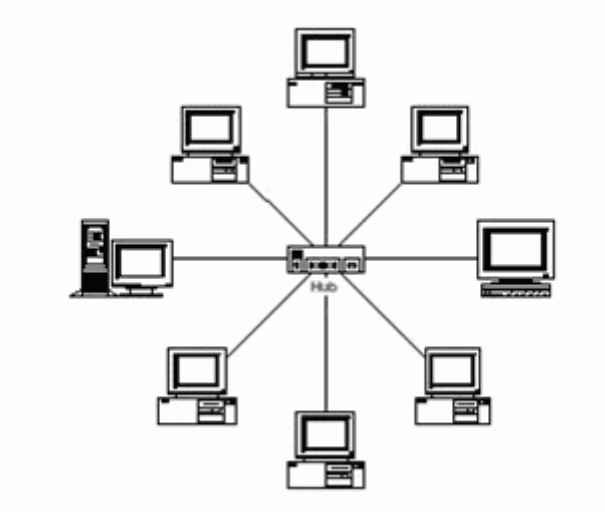


Gambar 2.17 Topologi Ring

3. Topologi Star

Pada tipe jaringan ini setiap workstation dihubungkan ke servermenggunakan suatu konsentrator. Masing-masing workstation tidak saling berhubungan. Jadi setiap user yang terhubung ke server tidak akan dapat berinteraksi dan melakukan apa-apa sebelum komputer server kita hidupkan. Apabila komputer server kita matikan, maka semua koneksi

jaringan akan terputus. Jaringan ini cocok untuk digunakan dalam pengembangan.



Gambar 2.18 Topologi Star

2.2.12 UML (*Unifield Modeling Language*)

Pemodelan (*modeling*) adalah gambaran dari realita yang dituangkan dalam bentuk pemetaan dengan aturan tertentu. Permodelan dalam dunia pembangunan perangkat lunak digunakan dalam proses merancang perangkat lunak sebelum melakukan pengkodean (*coding*) tujuannya untuk mempermudah langkah berikutnya dari pembangunan perangkat lunak sehingga lebih terencana dan semua kebutuhan pengguna dapat terpenuhi dengan lengkap dan tepat, termasuk faktor-faktor seperti *scalability*, *robustness*, *security*, dan sebagainya.

Kesuksesan suatu pemodelan perangkat lunak ditentukan oleh tiga unsur, yang kemudian terkenal dengan sebuan segitiga sukses (*the triangle for success*). Ketiga unsur tersebut adalah metode pemodelan (*notation*), proses (*process*) dan *tool* yang digunakan. Memahami notasi pemodelan tanpa mengetahui cara

pemakaian yang sebenarnya (proses) akan membuat sebuah proyek gagal. Dan pemahaman terhadap metode pemodelan dan proses disempurnakan dengan penggunaan *tool* yang tepat.

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan men-dokumentasikan sistem perangkat lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. UML kita dapat membuat model untuk semua jenis perangkat lunak, dimana aplikasi tersebut dapat berjalan pada perangkat keras, sistem operasi dan jaringan serta ditulis dalam bahasa pemrograman. Tetapi karena UML menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan perangkat lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural seperti C, Delphi atau VB.

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram perangkat lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*).

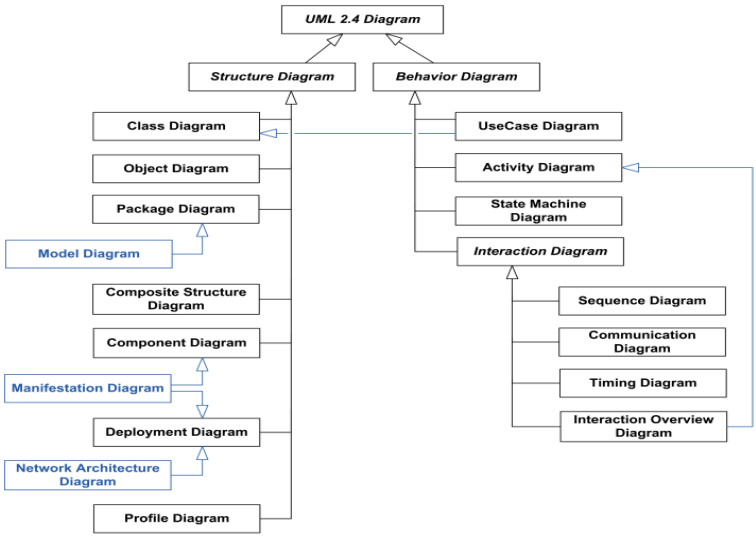
Dimulai pada bulan Oktober 1994 Booch, Rumbaugh dan Jacobson, yang merupakan tiga tokoh yang metodologinya banyak digunakan mempelopori usaha

untuk penyatuan metodologi perancangan berorientasi objek. Pada tahun 1995 dirilis draft pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh Object Management Group (OMG). Tahun 1997 UML versi 1.1 muncul, dan hingga saat ini telah ada versi 2.4.1 yang terdiri dari 4 macam spesifikasi, yaitu *Diagram Interchange Sfesification*, UML *Infratructure*, UML *Super Structure* dan *Object Constraint Language* (OCL). Seluruh spesifikasi tersebut dapat diakses di http://www.omg.org

2.2.10.1 Diagram UML

Pada UML versi 2.4 terdapat 2 kategori diagram sebagai berikut:

1. *Structure Diagram* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. *Behaviour Diagram* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.



Gambar 2.30 Diagram UML

2.2.10.2 *Class Diagram*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan perancangan berorientasi objek. *Class* menggambarkan keadaan (atribut) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda).

Sebuah Kelas memiliki struktur sebagai berikut:

1. Nama Kelas/Stereotype
2. Atribut, yaitu variabel-variabel yang dimiliki suatu kelas.
3. Metoda, yaitu fungsi-fungsi yang dimiliki oleh suatu kelas.

Class Diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem.

Diagram Class mempunyai 3 macam *relationalships* (hubungan), sebagai berikut :

1. *Association*

   Assosiation adalah 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.

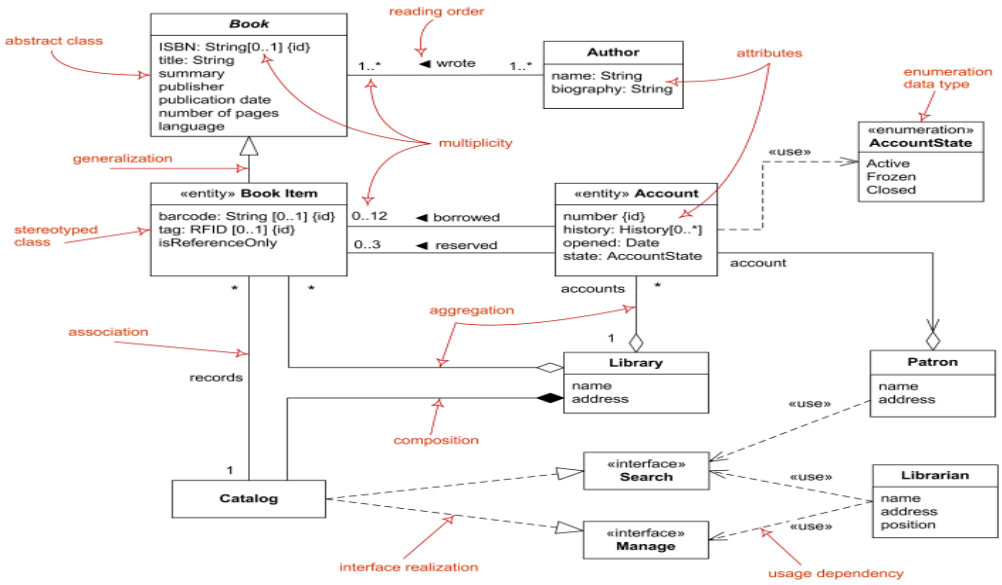
2. *Aggregation*

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

3. *Generalization*

Generalization Suatu hubungan turunan dengan mengasumsikan satu kelas merupakan suatu *Super Class* dari kelas yang lain. *Generalization* memiliki tingkatan yang berpusat pada *Super Class*.

Berikut ini adalah gambar contoh *class diagram* perpustakaan:

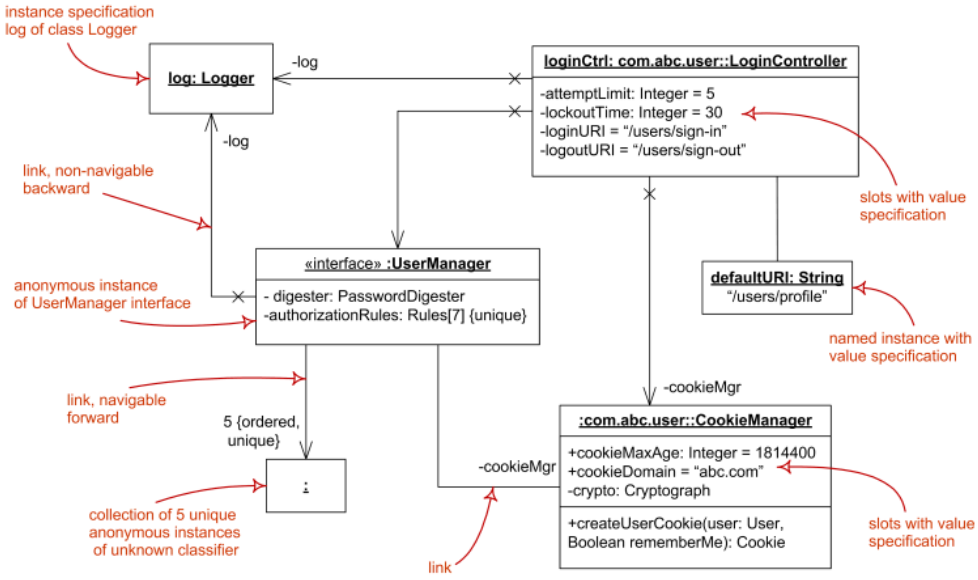


Gambar 2.31 Contoh *Class Diagram*

2.2.10.3 *Object Diagram*

Object diagram menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem. Pada *object diagram* harus dipastikan semua kelas yang sudah didefinisikan pada diagram kelas harus terpakai objeknya.

Contoh *object diagram*:



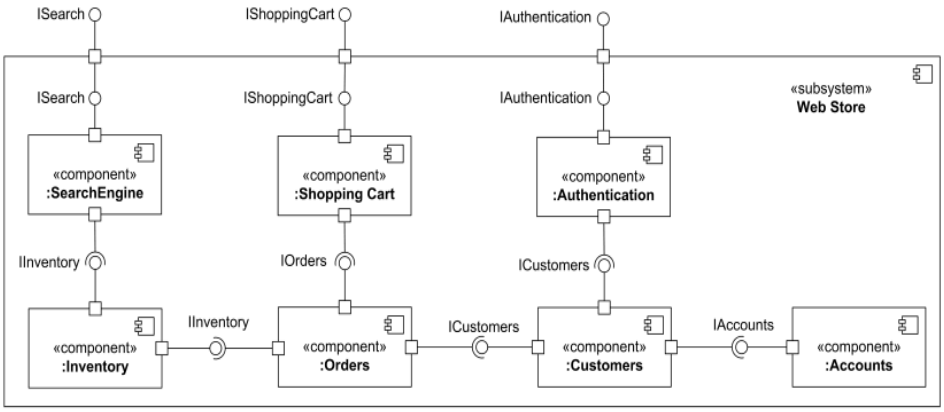
Gambar 2.32 Contoh *Object Diagram*

2.2.10.4 *Component Diagram*

Component diagram menggambarkan struktur dan hubungan antar komponen perangkat lunak, termasuk ketergantungan (*dependency*) di antaranya.

Komponen perangkat lunak adalah modul berisi code, baik berisi source code maupun binary code, baik library maupun executabel, baik yang muncul pada compile time, link time, maupun run time. Umumnya komponen terbentuk dari beberapa class dan/atau package, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa interface, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

Contoh Component Diagram:



Gambar 2.33 Contoh Component Diagram

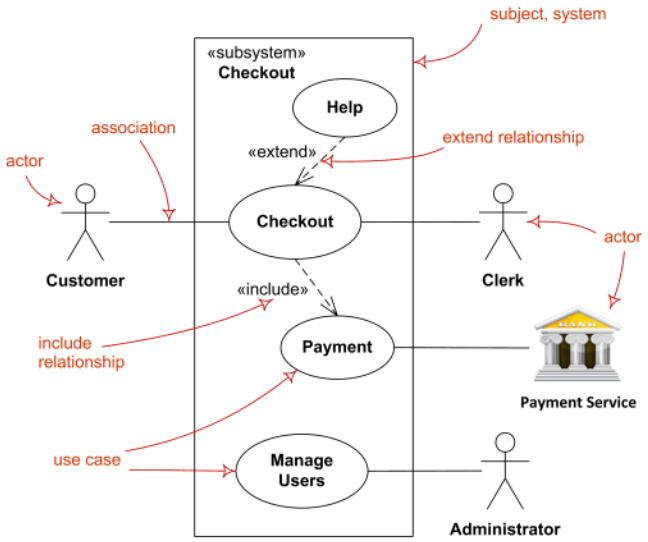
2.2.10.5 *Use Case Diagram*

Use case diagram merupakan permodelan untuk menggambarkan kelakuan (*behavior*) sistem yang akan dibangun. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibangun. *use case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut.

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkindan dapat dipahami. Ada dua hal utama pada *use case* yaitu aktor dan *use case.*

1. Aktor merupakan orang, proses atau sistem lain yang berinteraksi dengan sistem yang akan dibangun.
2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antarunit atau actor

Contoh *use case diagram*:



Gambar 2.37 Contoh *Use Case*

2.2.10.6 *Activity Diagram*

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

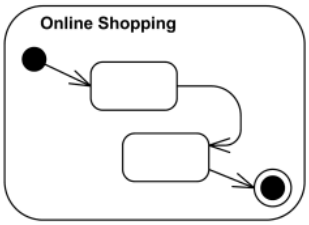
Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu, *activity diagram* tidak menggambarkan *behaviour internal* sebuah sistem secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case*

menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

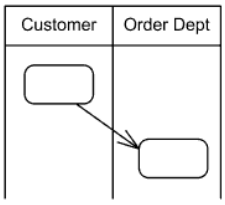
Seperti halnya *state*, standar UML menggunakan segi empat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal.

Contoh *activity diagram* belanja online:



Gambar 2.38 Contoh *Aktivity Diagram*

Activity diagram dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.



Gambar 2.39 Object Swimelane

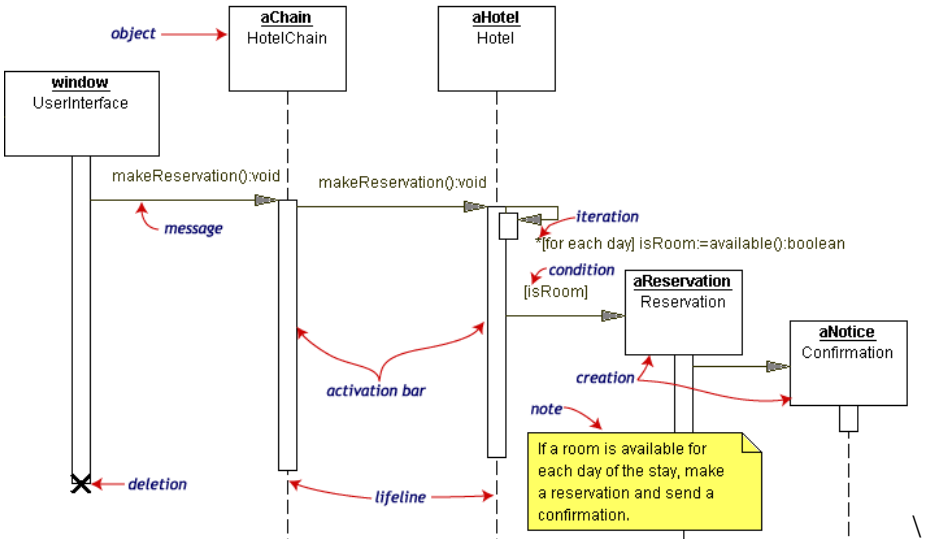
2.2.10.7 *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem berupa pesan yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah kejadian untuk menghasilkan output tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*.

Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan simbol khusus untuk *objek boundary*, *controller* dan *persistent entity*.

Contoh sequence diagram:



Gambar 2.41 Contoh *Sequence Diagram*

2.2.10.8 *Interaction Overview Diagram*

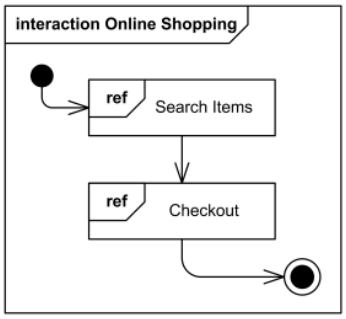
Interaction overview diagram mirip dengan activity diagram yang berfungsi untuk menggambarkansekumpulan urutan aktivitas. *Interaction overview diagram* adalah bentuk aktivitas diagram yang setiap titik merepresentasikan diagram interaksi. Interaksi diagram dapat meliputi *sequence diagram, communication diagram, timming diagram* dan *interaction overview diagram.*

Hampir semua interaksi pada *Interaction overview diagram* sama dengan notasi pada *activity diagram*. Tambahan pada *interaction overview diagram* adalah *interaction occurence* dan *interaction element*.

1. *Interaction Occurence* atau kejadian interaksi adalah referensi yang digunakan untuk diagram interaksi yang ada
2. *Interaction Eelement* atau elemen interaksi mirip *interaction occurence* perbedaannya didalam interaction element menampilkan isi diagram yang

direfrensikan secara langsung. Sedangkan *interaction occurence* hanya menampilkan nama diagram yang direfrensikan.

Contoh *interaction overview diagram*:

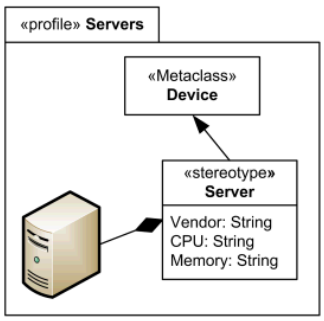


Gambar 2.44 Contoh Interaction overview diagram

2.2.10.9 *Profil Diagram*

Profil diagram menggambarkan mekanisme ekstensi UML dengan mendefinisikan stereotype, tagged value dan constraint. Profil memungkinkan adaptasi dari model UML yang berbeda platform dan domain.

Contoh *profil diagram* server:



Gambar 2.45 Contoh *Profil diagram*

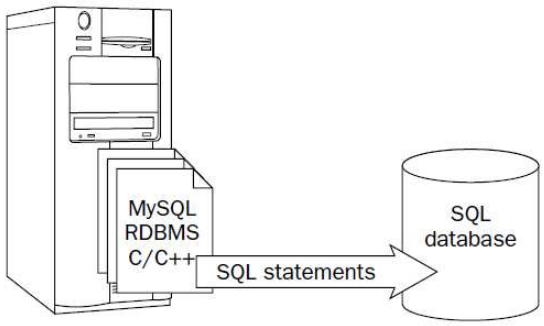
2.2.13 MySQL

MySQL adalah sebuah produk *Relational Database Management Sistem* (RDBMS). MySQL dapat mengakses data, memanipulasi data, melindungi data dari korupsi dan inkonsistensi serta mempertahankan metadata yang diperlukan untuk menentukan data yang disimpan dalam *database*. Perbedaan antara DBMS dengan RDBMS adalah bukan hanya dari mendukung atau tidaknya penyimpanan data dalam tabel, tetapi juga hubungan antartabel. MySQL menyediakan banyak fitur yang mendukung sebuah lingkungan yang aman untuk menyimpan, memelihara, dan mengakses data. Berikut kelebihan-kelebihan yang ditemukan di MySQL:

1. Skalabilitas, MSQL dapat menangani *database* yang besar, yang telah dibuktikan implementasinya dalam organisasi seperti Yahoo, Google, Cisco, HP, NASA dan lain sebagainya.
2. Portabilitas, MySQL dapat berjalan pada berbagai macam sistem operasi termasuk Windows, Unix, Linux, Solaris dan Mac OS. Juga dapat berjalan pada arsitektur yang berbeda, mulai dari *low-end* PC sampai *high-end mainframe*.
3. Konektivitas, MySQL sepenuhnya mendukung jaringan dan dapat diakses dari mana saja di internet serta pengguna dapat mengakses *database* MySQL secara bersamaan. MySQL juga menyediakan berbabagai macam API (*Aplication Program interface*) untuk mendukung konektivitas aplikasi yang ditulis dalam bahasa C, C++, Perl, PHP, Java, Phyton, C# dan lain sebagainya.

4. Keamanan, MySQL mencakup seluruh keamanan yang kuat untuk mengontrol akses data dan juga mendukung *Secure Socket Layer* (SSL) *Protocol*
5. Kecepatan, MySQL dikembangkan dengan kecepatan.
6. Mudah digunakan, MySQL mudah untuk digunakan dan diimplementasikan.
7. *Open Source*, MySQLAB membuat kode MySQL tersedia untuk digunakan setiap orang. filosofi *open source* memungkinkan khalayak global untuk berpartisipasi dalam pengembangan

Sebuah DBMS/RDBMS tidak dapat lepas dari SQL (*Structured Query Language*). SQL merupakan sebuah bahasa yang digunakan untuk mengelola dan berinteraksi dengan data dalam *database* relasional. SQL adalah bahasa *database* yang paling universal digunakan, dan itu telah menjadi bahasa standar untuk manajemen *database*. SQL bekerja sama dengan sebuah RDBMS untuk mendefinisikan struktur dari *database*, menyimpan data di *database* tersebut, memanipulasi data, mengambil data, mengontrol akses ke data, dan menjamin integritas data.



Gambar 2.46 Interaksi SQL dengan MySQL RDBMS

Beberapa perintah dasar SQL yang sering dipergunakan pada MySQL:

1. *Create Database*, perintah yang digunakan untuk membuat database baru.
   Sintaks : CREATE DATABASE DATABASE_NAME
2. *Drop Database*, perintah yang digunakan untuk menghapus database.
   Sintaks : DROP TABEL TABEL_NAME
3. *Create Tabel*, perintah yang digunakan untuk membuat tabel baru.
   Sintaks : Create Tabel tabel_name (create_definition)
4. *Describe*, perintah yang digunakan untuk mendeskripsikan tabel.
   Sintaks : Describe (Desc) tabel [colum]
5. *Alter Tabel*, perintah yang digunakan untuk memodifikasi tabel.
   Sintaks : Alter [Ignor] Tabel tabel_name
6. *Drop Tabel*, perintah yang digunakan untuk menghapus tabel.
   Sintaks : Drop Tabel tabel_name [tabel_name..]
7. *Delete*, perintah yang digunakan untuk menghapus record dri tabel.
   Sintaks : Delete From tabel_name Where Where_definiition
8. *Select*, perintah yang digunakan untuk query ke database.
   Sintaks : select * from tabel_name

MySQL mendukung beberapa API yang memungkinkan aplikasi yang ditulis dalam berbagai jenis bahasa pemrograman untuk berkomunikasi dengan database MySQL. Berikut beberapa API yang didukung oleh MySQL:

1. C API, adalah antarmuka pemrograman utama yang memungkinkan aplikasi C/C++ untuk menghubungkan ke MySQL. Sebagian besar

aplikasi klien termasuk dalam distribusi MySQL yang ditulis dalam C dan bergantung pada API ini.

2. ODBC, MySQL mendukung Open Database Connectivity (ODBC) melalui MySQL Connector / ODBC. ODBC adalah konektivitas database standar yang memungkinkan berbagai jenis aplikasi untuk menghubungkan ke berbagai jenis database.
3. JDBC, MySQL mendukung Java Database Connectivity (JDBC) melalui MySQL Connector / JDBC. JDBC adalah konektivitas *database* standar yang memungkinkan Java untuk menghubungkan ke berbagai jenis database.
4. PHP API, yang sekarang termasuk dengan *preprocessor* PHP, memungkinkan *script* PHP pada halaman Web untuk berkomunikasi secara langsung dengan *database* MySQL. Koneksi *database* dan permintaan data (melalui pernyataan SQL) dikodekan langsung dalam *script* PHP.

2.2.14 NET Framework Bahasa Pemrograman C#

1. .NET Framework

Microsoft mengumumkan C# pada tahun 2000 bersamaan dengan pengumuman *platform* .NET Framework. .NET Framework adalah *platform* yang merupakan perwujudan teknologi modern. Framework .NET diciptakan untuk dapat memecahkan masalah yang banyak dihadapi dunia pemrograman secara lebih efisien. Metodologi perancangan

berorientasi objek dan konsep software sebagai komponen tertanam dengan kuat di Framework.NET.

2. Bahasa Pemrograman C#

C# adalah bahasa pemrograman berbasis objek yang didukung oleh Microsoft .NET Framework. C# dibuat dengan tujuan untuk menyediakan bahasa pemrograman yang sederhana, modern, berorientasi objek, berpusat pada internet dan memiliki kinerja tinggi.[1] C# adalah salah satu dari banyak bahasa yang bisa dipakai untuk pemrograman .NET. Kelebihan utama bahasa ini adalah sintaksnya yang mirip C++, Java dan VB namun lebih sederhana dan mudah. Untuk mencapai kemudahan inilah konsep-konsep sulit pada C++, Java dan VB disederhanakan dengan mencakup semua dukungan dukungan untuk pemrograman terstruktur, pemrograman berbasis komponen dan pemrograman berorientasi objek.

C# dikembangkan oleh sebuah tim yang dipimpin oleh Anders Hejlsbreg dan Scott Wiltamuth yang sebelumnya dikenal karena menciptakan Turbo Pascal dan merancang Borland Delphi.

Jantung dari setiap pemrograman berorientasi objek adalah adanya dukungan untuk mendefinisikan dan bekerja dengan kelas. Kelas mendefinisikan tipe baru yang memungkinkan programmer untuk memperpanjang bahasa. C# mengandung keyword untuk mendeklarasikan kelas baru, metode dan properti serta sebagai bahasa yang berorientasi objek C# juga dapat mengimplementasikan enkapsulasi, pewarisan dan polimorfisme[12].

C# sebagai bahasa pemrograman untuk Framework .NET memiliki ruang lingkup penggunaan yang sangat luas seperti pembuatan perangkat lunak dengan *user interface* Windows maupun *console* dan karena Framework .NET memberikan fasilitas untuk berinteraksi dengan kode yang *unmanaged*, *library* seperti DirectX dan OpenGL dapat digunakan di C#. C# juga dapat digunakan untuk pemrograman *web site*, *web service*, membangun perangkat lunak Zune, perangkat lunak permainan (desktop dan XBOX), perangkat lunak *mobile* dan lain sebagainya.

2.2.15 Visual Studio

Microsoft Visual Studio merupakan sebuah perangkat lunak yang dapat digunakan untuk melakukan pengembangan aplikasi, baik aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya dalam bentuk aplikasi console, aplikasi Windows, ataupun aplikasi Web. Visual Studio mencakup kompiler, SDK, *Integrated Development Environment* (IDE), dan *Library*. Kompiler yang dimasukkan ke dalam paket Visual Studio antara lain Visual C++, C#, VB.NET, J++ dan J#.

Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam *unmanaged*/*native code* (dalam bentuk bahasa mesin yang berjalan di atas Windows) ataupun *managed* code (dalam bentuk Microsoft Intermediate Language di atas .NET Framework). Selain itu, Visual Studio juga dapat digunakan untuk mengembangkan aplikasi Silverlight, aplikasi Windows Mobile (yang berjalan di atas .NET Compact Framework).

Rilis terakhir Microsoft Visual Studio adalah Versi 2010 dan mendukung platform .NET Framework 4.0 dan versi terdahulunya. versi sebelumnya Visual Studio 2008 yang ditujukan untuk platform .NET Framework 3.5, Visual Studio 2005 ditujukan untuk platform .NET Framework 2.0 dan 3.0. Visual Studio 2003 ditujukan untuk .NET Framework 1.1, dan Visual Studio 2002 ditujukan untuk .NET Framework 1.0. Versi-versi tersebut di atas kini dikenal dengan sebutan Visual Studio .NET, karena membutuhkan Microsoft .NET Framework. Sementara itu, sebelum muncul Visual Studio .NET, terdapat Microsoft Visual Studio 6.0

Beberapa keuntungan Visual Studio:

1. Dapat membuat aplikasi untuk beberapa platform termasuk Windows 7, Windows Server 2008 R2, SQL Server 2008, SharePoint. Office, Windows Azure, dan Windows Phone.
2. Lingkungan pemrograman dapat dipersonalisasi
3. Dukungan *Application Lifecycle Management* dapat memudahkan kolaborasi, melakukan *testing* ataupun *debugging*.

2.2.16 WAMP

WAMP (*Windows and the principal components of the package: Apache, MySQL and PHP*) merupakan paket *web server* terdiri dari *Apache*, *database* MySQL dan PHP (*Hypertext Preprocessor*) yang bekerja pada localhost dan dibuat secara independen untuk digunakan pada sistem operasi Windows.

Beberapa kelebihan WAMP:

1. Menyediakan layanan web server yang bekerja secara *offline* (bekerja pada localhost).
2. Menyediakan layanan database MySQL secara *offline*.
3. Menyediakan layanan akses terhadap bahasa scripting PHP secara *offline*.
4. Menyediakan antarmuka pengguna berbasis web untuk mengelola database MySQL dengan adanya fitur phpMyAdmin dan SQLiteManager.