BAB II

LANDASAN TEORI

2.1 Konsep Dasar Sistem Informasi

Bagian ini akan memaparkan tentang konsep dasar dari sistem informasi yang akan dibahas dalam beberapa sub bab.

2.1.1 Definisi Sistem

Menurut [**Gbd94**] sistem dapat dibagi menjadi dua yaitu sistem abstrak dan sistem fisik sistem adalah “suatu susunan teratur gagasan atau konsepsi yang saling tergantung” sedangkan sistem fisik dilihat dari contohnya seperti sistem peredaran darah, sistem transportasi, sistem komputer dapat diartikan “sebuah sistem yang terdiri dari bagian - bagian yang saling berkaitan yang beroperasi bersama untuk mencapai beberapa sasaran atau maksud”. Dengan demikian sebuah sistem bukanlah seperangkat unsur yang tersusun secara tak teratur, tetapi terdiri dari unsur yang dapat dikenal sebagai saling melengkapi sarena suatu maksud, tujuan, atau sasaran. Sistem fisik lebih dari sekedar bentuk konseptual, karena dapat memperlihatkan kegiatan atau prilaku.

Sedangkan menurut [**Jog99**] “sistem dapat didefinisikan sebagai suatu kesatuan yang terdiri dari dua atau lebih komponen atau subsistem yang berinteraksi untuk mencapai suatu tujuan”.

2.1.2 Definisi Informasi

Menurut [**Gbd94**] Definisi umum untuk informasi dalam pemakaian sistem informasi “adalah suatu data yang telah diolah menjadi sebuah bentuk yang

lebih berguna dan lebih berarti bagi si pemakai dalam mengambil suatu keputusan sekarang atau keputusan yang akan datang".

Menurut [**Jog99**] Kualitas dari informasi yang dihasilkan haruslah berisikan tiga hal berikut, yaitu :

1. Informasi harus akurat, artinya informasi harus jelas dan bebas dari kesalahan-kesalahan.
2. Informasi harus tepat waktu, artinya informasi yang disampaikan harus tepat waktu dan tidak boleh terlambat.
3. Informasi harus relevan, artinya informasi tersebut mempunyai manfaat untuk pemakainya.

Suatu informasi dapat mempunyai nilai. Nilai dari informasi dapat ditentukan dengan dua hal, yaitu manfaat dan biaya mendapatkannya. Suatu informasi dikatakan bernilai bila manfaatnya lebih efektif dibandingkan dengan biaya mendapatkannya.

2.1.3 Definisi Sistem Informasi

Menurut [**Gbd94**] "Sistem informasi bagi setiap perusahan atau instansi swasta maupun negara adalah suatu kebutuhan yang tidak bisa dipisahkan, karena dengan adanya penerapan sistem informasi yang dikembangkan dengan baik di kalangan lembaga negara atau perusahaan swasta tentunya dapat menjamin kelangsungan hidup organisasinya".

Modul sistem terdiri dari empat elemen subsistem:

1. Masukan
2. Pengolahan

3. Keluaran
4. Umpan balik/kontrol

Perumpamaan pada perusahaan produksi barang masukannya adalah bahan mentah, pengolahannya dilakukan dengan menggunakan fasilitas pabrik, keluarannya adalah barang jadi, dan umpan baliknya adalah terdiri dari usul perbaikan yang diberikan oleh unit pengawasan mutu dari perusahaan bersangkutan atau dari para pelanggan.

Ada yang mengemukakan bahwa sistem itu ada yang disebut:

Sistem Terbuka adalah sistem yang bekerjanya dapat dipengaruhi oleh faktor - faktor luar contohnya sistem cuaca, sistem peredaran darah, sistem komputer, dll.

Sistem Tertutup adalah sistem yang bekerjanya tidak dipengaruhi oleh faktor - faktor luar contohnya sistem pekerjaan laboratorium untuk proses tabung kimia.

Pada pekerjaan informasi yang diutamakan adalah pembagian sistem yang berdasarkan karakteristik sistem fisik dan sistem nonfisik. Pada sistem fisik yang diolah adalah bentuk fisik. Sedangkan pada sistem nonfisik yang diolah adalah sesuatu yang abstrak misalnya buah pikiran, pendapat, perhitungan, huruf - huruf, angka - angka, dan lain - lain yang ke semuanya bukan barang nyata.

Modul sistem informasi terdiri dari subsistem data, pengolahan, informasi, dan umpan balik.

Pembangunan atau pengembangan sistem informasi dimulai ketika disadari bahwa sistem informasi yang telah ada sekarang sudah tidak memenuhi persyaratan lagi atau yang lebih tepatnya perlu penyempurnaan lagi dari yang sebelumnya untuk dapat menjadikan sistem tersebut menjadi yang lebih baik.

Disini ada beberapa langkah yang harus ditempuh dalam pengembangan sistem informasi, diantaranya adalah:

1. Studi Kelayakan

Tujuannya adalah untuk mendapatkan gambaran secara konkret dari sistem yang berjalan sekarang. Dalam tahap ini menyangkut 3 kegiatan, yaitu :

a. Mengklarifikasi permintaan.
b. Mengklarifikasi studi kelayakan.
c. Persetujuan terhadap studi kelayakan dan manajemen.

Studi kelayakan berisi informasi dengan ruang lingkup dimana sistem akan diimplementasikan, batasan yang mempengaruhi proyek, usulan alternatif tersebut. Studi kelayakan juga menerangkan deskriptif bentuk sistem yang diinginkan beserta analisa untung ruginya dan evaluasi resiko teknik.

2. Pendefinisian Kebutuhan Sistem

Menentukan kebutuhan adalah proses mempelajari sistem yang berjalan dan mencari kekurangan untuk melakukan suatu perbaikan. Sistem yang berjalan dapat dianggap terdiri dari berbagai proses yang berinteraksi untuk pencapaian hasil tertentu. Jadi tiap proses dianggap sebagai elemen dari sistem tersebut, yaitu mencari dan mendeteksi proses yang ada, berikut segala masalah dari struktur analisa sistem yang berjalan.

Tujuan dari pendeteksian ini adalah untuk mendapatkan spesifikasi fungsi yang akan dirancang. Dan setelah itu dilanjutkan dengan mendesain eksternal sistem yang relevan dengan fungsi yang dirancang, seperti format dan screen yang dibutuhkan, format report yang dihasilkan, dan lain-lain.

3. Desain Sistem

Tahap pendesainan sistem terdiri dari desain awal dan pendesainan terperinci. Desain awal dapat didefinisikan sebagai penggambaran, perancangan pembuatan sketsa dan penyusunan beberapa elemen data. Dalam tahap desain harus dipertimbangkan bagaimana sistem ini bekerja. Dalam mendesain sistem digunakan 3 alternatif dasar, yaitu:

1. Tidak mengubah sistem yang sedang berjalan .
2. Memodifikasi sistem yang sedang berjalan.
3. Merubah sistem yang sedang berjalan

Untuk langkah-langkah dasar dalam mendesain secara garis besar adalah sebagai berikut :

1. Menetapkan tujuan sistem.
2. Menentukan peranan dari sistem yang dirancang dari sistem secara keseluruhan.
3. Mengembangkan model logika berdasarkan
4. Penentuan general sistem desain, yaitu menentukan bagaimana data atau bahan yang telah dikumpulkan dimasukkan dalam sistem yang baru.

5. Tahap Implementasi Sistem.

   Tahap ini dilakukan setelah desain selesai dibuat. Setelah dianggap lengkap dan baik maka selanjutnya menerapkan sistem tersebut sesuai dengan prosedur dan proses yang telah ditetapkan pada rancangan sistem. Implementasi adalah tahap pemakaian sistem, termasuk user training dan pembuatan file-file yang dibutuhkan oleh sistem tersebut.

Menurut [**Gbd94**] Bahwa informasi merupakan hal yang sangat penting bagi manajemen didalam pengambilan keputusan. Untuk pengertian suatu sistem informasi banyak juga yang mendefinisikan, diantaranya adalah :

1. Sistem informasi adalah suatu sistem didalam organisasi yang mempertemukan kebutuhan pengolahan data (kejadian), mendukung operasi atau proses, menyediakan laporan-laporan atau dokumen yang diperlukan.
2. Sistem Informasi adalah sekumpulan prosedur orgnisasi yang pada saat dilaksanakan akan memberikan informasi bagi pengambil keputusan (*Decision Maker*) dan atau untuk mengendalikan organisasi.
3. Sistem Informasi adalah hasil dari hubungan yang erat dari sistem dan informasi, maka sistem informasi dapat disimpulkan sebagi suatu sistem yang terdiri dari berbagai proses kerja sub-sub sistem yang menghasilkan informasi sebagi bahan pengambilan keputusan untuk mencapai tujuan tertentu yang telah ditetapkan.
4. Sistem Informasi adalah suatu sistem yang berisi himpunan yang terintegrasi dari komponen-komponen manual dan komponen - komponen

terkomputerisasi yang bertujuan untuk mengumpulkan data dan menghasilkan informasi untuk pemakai

2.2 Definisi P.O. Bhinneka Sangkuriang

PO Bhinneka Sangkuriang yang di kenal sebagai perusahaan jasa angkutan antar kota yang mampu memenuhi fasilitas pelayanan angkutan yang di kehendaki oleh masayarakat. Dengan memberi kepuasan dalam bentuk perhatian pelayanan secara khusus dan pribadi, maka dalam melakukan bisnisnya perlu adanya sistem informasi yang dapat memudahkan dalam setiap kinerja baik dari perusahaan maupun karyawan.

Sistem informasi ini akan mendukung kinerja perusahaan yang bersangkutan baik dalam pelayanan terhadap masyarakat sampai karyawan. Namun sesudah perkembangan teknologi informasi yang sedemikian pesatnya, perusahaan harus setiap saat mengupdate sistem informasinya sehingga dalam kinerjanya akan menuju ketitik yang lebih baik.

2.3 Arsitektur Aplikasi

Bagian ini akan memaparkan tentang arsitektur jaringan yang akan dibahas dalam beberapa sub bab.

2.3.1 Pengertiaan Jaringan Komputer

Jaringan komputer adalah sekelompok komputer otonom yang dihubungkan satu dengan yang lainnya dengan menggunakan protokol komunikasi melalui media transmisi atau media komunikasi sehingga dapat saling berbagi data informasi, program – program, penggunaan bersama perangkat keras seperti *printer* atau *harddisk.* [**Irb02**] Prinsip dasar dalam sistem jaringan ini

adalah proses pengiriman data atau informasi dari pengirim ke penerima melalui suatu media komunikasi tertentu.

2.3.2 Jenis - Jenis Jaringan Komputer

Jaringan komputer yang digunakan dewasa ini dapat dikelompokkan berdasarkan luasan area yang dapat dijangkau. Luasan area ini pada mulanya ditentukan berdasarkan jarak jangkau dalam satuan meter. Namun dalam perkembangan selanjutnya berdasarkan peralatan dan fasilitas. Penggunaan peralatan – peralatan yang dimaksud adalah peralatan – peralatan *inter* jaringan seperti *Repeater, Router* atau *Gateway* dari peralatan tersebut dapat dilihat jangkauan area jaringan dan luasan segmen jaringan yang dibangun. Berdasarkan kriteria – kriteria tersebut secara umum jaringan komputer dapat digolongkan sebagai berikut :

2.3.2.1 *Local Area Network* (LAN)

Sebuah LAN adalah jaringan yang dibatasi oleh area yang relatif kecil, umumnya dibatasi oleh area lingkungan seperti sebuah perkantoran di sebuah gedung, atau sebuah sekolah, dan biasanya tidak jauh dari sekitar 1 km persegi. Beberapa model konfigurasi LAN, satu komputer biasanya menjadi sebuah *file server,* digunakan untuk menyimpan perangkat lunak (*software*) yang mengatur aktifitas jaringan, ataupun sebagai perangkat lunak yang dapat digunakan oleh komputer - komputer yang terhubung ke dalam *network*. Komputer - komputer yang terhubung kedalam jaringan itu biasanya disebut dengan *workstation.* Biasanya kemampuan *workstation* lebih di bawah dari *file server* dan mempunyai aplikasi lain di dalam *harddisk*nya selain aplikasi untuk jaringan. Kebanyakan

LAN menggunakan media kabel untuk menghubungkan antara satu komputer dengan komputer lainnya.

2.3.2.2 *Metropolitan Area Network* (MAN)

Sebuah MAN, biasanya meliputi area yang lebih besar dari LAN, misalnya antar wilayah dalam satu provinsi. Dalam hal ini jaringan menghubungkan beberapa buah jaringan - jaringan kecil ke dalam lingkungan area yang lebih besar, sebagai contoh yaitu : jaringan Bank dimana beberapa kantor cabang sebuah Bank di dalam sebuah kota besar dihubungkan antara satu dengan lainnya.

2.3.2.3 *Wide Area Network* (WAN)

Wide Area Networks (WAN) adalah jaringan yang lingkupnya biasanya sudah menggunakan sarana satelit ataupun kabel bawah laut sebagai contoh keseluruhan jaringan suatu Bank yang ada di Indonesia ataupun yang ada di negara - negara lain. menggunakan sarana WAN, Sebuah Bank yang ada di Bandung bisa menghubungi kantor cabangnya yang ada di Jepang, hanya dalam beberapa menit. Biasanya WAN agak rumit dan sangat kompleks, menggunakan banyak sarana untuk menghubungkan antara LAN dan WAN ke dalam komunikasi global seperti Internet. Tapi bagaimanapun juga antara LAN, MAN dan WAN tidak banyak berbeda dalam beberapa hal, hanya lingkup areanya saja yang berbeda satu diantara yang lainnya.

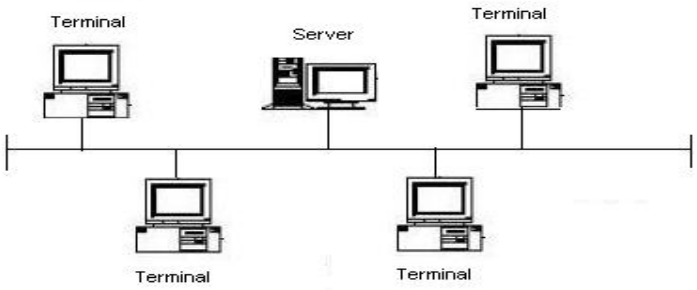
2.3.3 Topologi Jaringan Komputer

“Topologi adalah suatu cara menghubungkan komputer yang satu dengan komputer lainnya sehingga membentuk jaringan” menurut **[Irb02]**. Cara yang saat

ini banyak digunakan adalah *bus, token-ring, star*, dan *tree* Masing-masing topologi ini mempunyai ciri khas, dengan kelebihan dan kekurangannya sendiri.

2.3.3.1 Topologi BUS

Pada topologi Bus digunakan sebuah kabel tunggal atau kabel pusat di mana seluruh *workstation* dan *server* dihubungkan, seperti terlihat pada gambar 2.1. Topologi Bus menggunakan *T - Connector* (dengan terminator 50 ohm pada ujung *network*), maka komputer atau perangkat jaringan lainnya bisa dengan mudah dihubungkan satu sama lain.

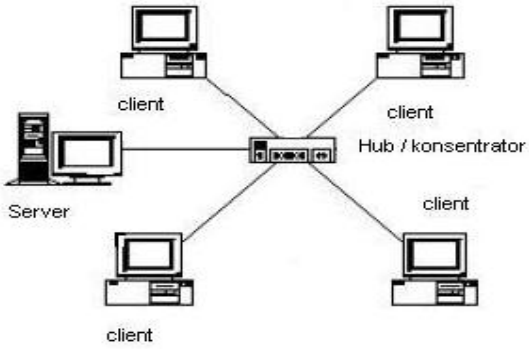


Gambar 2.1 Topologi Bus
(Sumber: Irawan, Budhi, (2002), *Diktat Kuliah Komunikasi Data*)

2.3.3.2 Topologi Star

Pada topologi *Star*, sebuah terminal pusat bertindak sebagai pengatur dan pengendali semua komunikasi data yang terjadi. Terminal – terminal lain terhubung ke terminal pusat dan pengiriman data dari satu terminal ke terminal lainnya melalui terminal pusat. Pada topologi *Star* digunakan Hub untuk menghubungkan terminal pusat dengan terminal – terminal lainnya, seperti terlihat pada gambar 2.2. Topologi ini mudah untuk dikembangkan, baik untuk

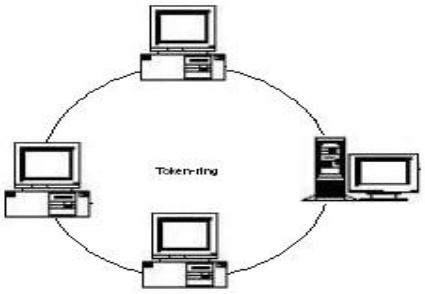
penambahan maupun pengurangan terminal. Banyaknya terminal yang dapat terhubung tergantung pada jumlah *port* yang tersedia pada Hub yang digunakan.



Gambar 2.2 Topologi Star
(Sumber: Irawan, Budhi, (2002), *Diktat Kuliah Komunikasi Data*)

2.3.3.3 Topologi Ring

Topologi ini memanfaatkan kurva tertutup, artinya informasi dan data serta *traffic* disalurkan sedemikian rupa ke masing - masing *node*. Umumnya fasilitas ini memanfaatkan *fiber optic* sebagai media penghubung jaringan. Didalam topologi *Ring* semua *workstation* dan *server* dihubungkan sehingga terbentuk suatu pola lingkaran atau cincin, seperti terlihat pada gambar 2.3. Tiap *workstation* ataupun *server* akan menerima dan melewatkan informasi dari satu komputer ke komputer lain, bila alamat - alamat yang dimaksud sesuai maka informasi diterima dan bila tidak, informasi akan dilewatkan.



Gambar 2.3 Topologi Ring

(Sumber: Irawan, Budhi, (2002), *Diktat Kuliah Komunikasi Data*)

2.3.4 Manfaat Jaringan Komputer

Tujuan dibangunnya suatu jaringan komputer adalah untuk membawa data informasi dari sisi pengirim menuju penerima secara cepat dan tepat tanpa adanya kesalahan melalui media transmisi atau media komunikasi tertentu.

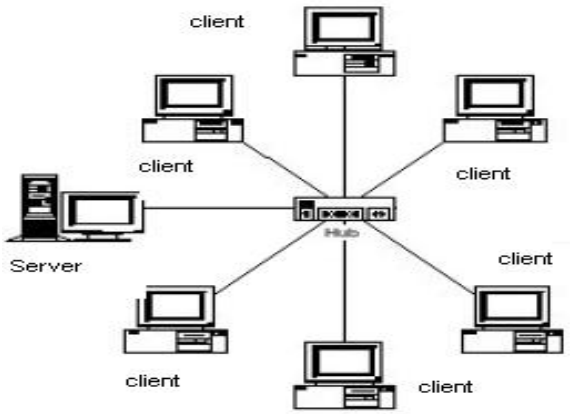
2.4 Pengertian Client Server

Arsitektur jaringan *Client Server* merupakan pengembangan dari arsitektur *file server*. Arsitektur ini adalah model konektivitas pada jaringan yang mengenal adanya *server* dan *client*, dimana masing - masing memiliki fungsi yang berbeda satu sama lain. Model ini merupakan pengembangan dari model *File Server*, karena pada model ini terdapat terminal khusus yang dapat melayani sampai pelayanan komputasi. Pada gambar 2.4 menunjukkan arsitektur jaringan *client server.*

Server adalah komputer yang menyediakan fasilitas bagi komputer - komputer lain di dalam jaringan dan *client* adalah komputer - komputer yang menerima atau menggunakan fasilitas yang disediakan oleh *server*. *Server* pada jaringan tipe *client-server* disebut dengan *Dedicated Server* karena murni

berperan sebagai *server* yang menyediakan fasilitas kepada *workstation* dan *server* tersebut tidak dapat berperan sebagai *workstation.*

Jadi *client-server* adalah merupakan konsep jaringan yang menghubungkan lebih dari satu komputer yang berperan sebagai *workstation* ke komputer pusat atau *server* yang memungkinkan untuk mengolah data secara bersama – sama.



Gambar 2.4 Jaringan Client-Server
(Sumber: Irawan, Budhi, (2002), *Diktat Kuliah Komunikasi Data*)

2.5 Aplikasi dan Teknologi Web

Bagian ini akan memaparkan tentang aplikasi dan teknologi web yang akan dibahas dalam beberapa sub bab.

2.5.1 Pengertian Internet

Menurut [**Irb02**] "Internet adalah suatu jaringan komputer global yang terbentuk dari jaringan - jaringan komputer lokal dan regional yang memungkinkan komunikasi data antar komputer yang terhubung ke jaringan tersebut"

Internet awalnya merupakan suatu recana dari Departemen Pertahanan Amerika Serikat (*US Departement of Defenses*) pada sekitar tahun 1960. Dimulai dari suatu proyek yang dinamakan ARPANET atau *Advanced Research Project agency Network*. Beberapa universitas di Amerika Serikat diantaranya UCLA, Stanford, UC Santa Barbara dan Univercity of Utah, diminta bantuan dalam mengerjakan proyek ini dan awalnya telah berhasil menghubungkan empat komputer di lokasi universitas yang berbeda tersebut. Perkembangan ARPANET ini cukup pesat dilihat perkembangan komputer pada saat itu.

Sebagai gambarannya pada tahun 1977 ARPANET telah menghubungkan lebih dari 100 mainframe komputer dan saat ini terdapat 4 juta *host* jaringan yang terhubung pada jaringan ini.

2.5.2 Istilah - Istilah Dalam *Internet*

Teknologi *internet* merupakan teknologi yang sudah tidak asing lagi dalam kehidupan pada masa sekarang ini dimana pemanfaatannya sudah banyak diterapkan pada berbagai bidang. Banyak istilah – istilah dalam *internet* diantaranya :

2.5.2.1 ISP (*Internet Service Provider*)

Menurut **[Irb02]** "ISP adalah suatu perusahaan atau badan baik pemerintah maupun swasta yang dapat memberikan fasilitas layanan koneksi ke jaringan Internet bagi jaringan yang terhubung kepadanya."

ISP yang ada pada saat ini merupakan dapat berupa ISP yang bersifat tertutup dan ISP yang bersifat umum. ISP yang bersifat tertutup adalah lembaga atau badan yang hanya melayani atau dapat memberikan fasilitas jaringan internet

kepada jaringan lokal dari lembaga atau badan tersebut. Contohnya terdapat pada beberapa departemen pemerintahan, lembaga penelitianm perusahaan, ataupun lembaga. ISP yang bersifat umum adalah sebuah perusahaan yang dapat memberikan layanan kepada masyarakat luas baik pribadi atau lembag yang ingin terkoneksi denagn jaringan internet. ISP seperti ini biasanya bersifat komersial, yaitu menjual jasa layanan koneksi dengan jaringan internet.

2.5.2.2 *Bandwidth*

Bandwidth atau kecepatan transmisi data, dalam satuan bit per detik.Semakin besar *bandwidth* maka semakin cepat transmisi datanya, contoh untuk komunikasi data dengan jaringan *Ethernet* mempunyai kecepatan 10-100 Mbps.

2.5.2.3 *Mail Server*

Mail sever adalah komputer yang berperan sebagai ‘Kantor Pos’ bagi sebuah jaringan. Fungsi kerjanya yaitu melayani suratkeluar dan surat masuk. Seperti pada contoh kasus jika seorang mengirin E-mail maka E-mail tersebut akan dikirimkan ke server ini.

2.5.2.4 *File Transfer Protokol* (*FTP*) *Server*

File Transfer Protokol (*FTP*) *Server* adalah komputer yang bertindak sebagai penampungn file-file yang dapat diakses oelh jaringan. File-file tersebut dapat di-*transfer* baik untuk disimpan pada file server dari jaringan (*upload*) ataupun diambil dari file server untuk ke komputer pada jaringan (*download*).

2.5.2.5 *Domain Name Service* (*DNS*) *Server*

Domain Name Service (*DNS*) *Server* adalah komputer yang berfungsi sebagai identitas nama domain yang dimilii oleh jaringan tersebut, dan juga memelihara informasi data dari domain yang lain.

2.5.2.6 *Electronic Mail* (E-Mail)

Adalah merupakan aplikasi pertama internet, E-mail ini merupakan sebuah surat atau pesan yang dapat dikirimkan melalui jaringan komputer ke antar komputer.

2.6 Perangkat Lunak Pendukung

Perangkat lunak pendukung adalah perangkat lunak yang digunakan untuk membangun sebuah perangkat lunak dalam hal ini penulis menggunakan beberapa perangkat lunak pendukung antara lain :

2.6.1 PHP (*Pheriperal Hyper Text Preprocessor*)

PHP secara resmi merupakan kependekan dari *Pheriperal Hyper Text Preprocessor*, merupakan bahasa *script server side* yang disisipkan pada HTML sedangkan menurut [**Bta04**] **"**PHP merupakan script untuk pemrograman 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 menggunakann editor text atau editor HTML".

PHP berawal saat Rasmus Lerdorf membuat sejumlah skrip *"Perl"* yang dapat mengamati siapa saja yang melihat - lihat daftar riwayat hidupnya, yakni pada tahun 1994. Skrip - skrip ini selanjutnya dikemas menjadi tool yang disebut

"Personal Home Page". Paket inilah yang menjadi cikal bakal PHP. Pada tahun 1995 , Rasmus menciptakan *PHP/FI* Versi 2. Pada Versi inilah pemrogram dapat menempelkan kode tersruktur di dalam tag *HTML (Hypertext Markup Language)* Kode PHP juga bisa berkomunikasi dengan database dan melakukan perhitungan-perhitungan yang kompleks sambil jalan.

Pada saat ini, PHP cukup popular sebagai piranti pemrograman Web, terutama di lingkungan Linux. Walaupun demikian, PHP sebenarnya juga dapat berfungsi pada server-server yang berbasis *UNIX*, *Windows NT*. PHP juga dirancang untuk diintegrasikan dengan *web server Apache.*

Dengan menggunakan PHP maka maintenance suatu situs web menjadi lebih mudah. Proses update data dapat dilakukan dengan menggunakan aplikasi yang dibuat dengan menggunakan script PHP, yang akhirnya software ini disebarkan dan dilisensikan sebagai perangkat lunak Open Source.

2.6.2 Mysql *Server*

Menurut [**Bta04**] **"**Mysql merupakan *software database* yang dikembangkan di lingkungan sistem operasi Linux yang menunjang performasi query dari databasenya yang saat ini bisa dikatakan paling cepat, dan jarang bermasalah"

2.7 Alat Bantu Pemodelan

Pemodelan adalah sebuah aktivitas yang telah berlangsung bertahun-tahun dalam pengembangan perangkat lunak. Sebuah model masih diperlukan untuk menulis sebuah aplikasi baik menggunakan bahasa pemograman yang paling

sederhana sampai bahasa yang kompleks. Pemodelan dapat berupa menggambar diagram alir (*flowchart*) yang menunjukkan langkat-langkah yang dikerjakan sebuah aplikasi. Pemodelan diperlukan untuk mendefinisikan sebuah model membuat mudah untuk memecah aplikasi yang komplek atau sebuah sistem yang besar menjadi sederhana, menjadi bagian-bagian diskrit yang mudah untuk dipelajari. Dengan alasan demikian, pemodelan harus memenuhi syarat:

1. *Readability*

   Readability membawa kejelasan – kemudahan dalam memahami. Memahami sistem adalah langkah pertama baik dalam membangun atau mengembangkan sebuah sistem yang telah ada. Hal ini melibatkan pengetahuan tentang bagaimana sistem dibuat, bagaimana sistem berperilaku dan sebagainya. Memodelkan sebuah sistem membuat sistem mudah dibaca dan yang paling penting, mudah dalam pendokumentasian. Penggambaran sebuah sistem agar mudah dibaca melibatkan pemahaman struktur sistem dan tingkah laku sistem.

2. *Reusability*

   Reusability adalah efek dari membuat sebuah sistem mudah dibaca. Bila sebuah sistem telah dimodelkan sedemikian rupa sehingga mudah dipahami, karena para pengembang cenderung untuk mencari kesamaan atau pengulangan, baik dalam hal fungsionalitas, fitur ataupun struktur.

2.7.1 *Unified Modeling Language* (UML)

UML mengintegrasikan konsep dari *Booch*, OMT, OOSE dan juga *Class-Relation* dengan menggabungkan mereka menjadi suatu kesatuan bahasa

modeling yang bisa berguna bagi siapa saja. UML bertujuan untuk menjadi standar bahasa modeling yang mampu untuk memodelkan sistem yang konkuren dan juga terdistribusi. UML sudah menjadi standar industri yang dibuat dibawah pengawasan *Object Management Group* (OMG). OMG ditugaskan untuk memberi informasi mengenai object-oriented methodology yang mampu untuk membuat bahasa untuk modeling software. Dengan menggunakan UML para pengembang dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa pemrograman berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VisualBasic atau C.

UML bukan merupakan metode dan juga tidak berusaha untuk menjadi suatu metode, namun demikian ia merupakan kebutuhan untuk *Rational Unified Process* yang telah dibuat oleh *Rational Software Corporation*. Metode development lainnya seperti *Dynamic Systems Development Method* juga menggunakan teknik dari UML.

UML menyediakan kemampuan untuk menangkap karakteristik sebuah sistem dengan menggunakan notasi-notasi tertentu. UML menyediakan sederetan notasi sederhana yang mudah dipahami untuk mendokumentasikan sistem berdasarkan prinsip-prinsip perancangan berorientasi objek.. UML sudah menjadi *standardized modelling language* yang terdiri dari kumpulan-kumpulan diagram,

dikembangkan untuk membantu para pengembang sistem dan software agar bisa menyelesaikan tugas-tugas seperti:

1. Spesifikasi
2. Visualisasi
3. Desain Arsitektur
4. Konstruksi
5. Simulasi dan testing
6. Dokumentasi

UML dikembangkan sebagai ide dasar untuk mempromosikan hubungan dan produktifitas antara para pengembang dari *object-oriented system.* Menurut **[Lar04]** ada tiga cara dalam memakai UML dalam melakukan pemodelan sistem:

1. UML sebagai sketsa

   UML digambarkan dalam sketsa coretan-coretan dalam kertas atau *whitboard* secara tidak formal. Biasanya digunakan dalam sesi diskusi tim untuk membahas aspek tertentu dalam tahap analisis dan perancangan.

2. UML sebagai *blueprint* system

   Seperti diagram kelistrikan adalah blueprint dari komponen atau produk yang akan dihasilkan, UML juga bisa menggambarkan blueprint yang identik untuk sebuah sistem software.

3. UML sebagai bahasa pemrograman

   UML berfungsi sebagai bahasa pemrograman mencoba melakukan semuanya dengan UML sampai kepada produk jadinya. Analisis dan perancangan dilakukan dengan diagram-diagram yang ada dalam UML**,** sementara sebuah tool atau generator bisa menghasilkan produk akhir dari diagram-diagram ini.

Saat ini UML paling banyak digunakan dengan cara pertama dan kedua. Khusus dalam metode *agile* (cepat dan ringan), UML digunakan dengan cara pertama.

2.7.2 OMG

Objek Manajemen Group (OMG) merupakan oerganisasi internasional yang didukung lebih dari 600 anggota, termasuk didalamnya perusahaan sistem informasi, pengembang perangkat lunak dn pengguna. OMG ditemukan pada tahun 1989, OMG mempromosikan teori dan praktek dari teknologi orieantasi objek dalam pengembangan perangkat lunak. Izin dari organisasi ini adalah untuk membangun sebuah spesifikasi manajemen objec dan garis haluan industri untuk melengkapi sebuah kerangka kerja biasa untuk sebuah pengembang aplikasi.

Tujuan utamanya adalah reusability, portability, and interoperability dari perangkat lunak yang berbasiskan objek dalam pendistribusiannya, ke tidak seragamannya, lingkungannya.

Sasaran OMG adalah mengadopsi dari perkembangan teknologi objek dan mempengaruhi setiap arahnya dengan membangun Objek Manajemen Arsitektur (OMA)

2.7.3 Model Diagram UML

UML terdiri atas banyak elemen-elemen grafis yang digabungkan membentuk diagram. Tujuan representasi elemen-elemen grafis ke dalam diagram adalah untuk menyajikan beragam sudut pandang dari sebuah sistem berdasarkan fungsi masing-masing diagram tersebut. Kumpulan dari beragam sudut pandang inilah yang disebut sebuah model. Pada saat ini UML telah mencapai versi 2.1

memiliki banyak perubahan dari versi sebelumnya seperti diagram kolobarasi diganti dengan diagram komunikasi dan terdapat tambahan tiga diagram. Penulis menggunakan UML versi 1.5 dan diagram-diagram yang resmi tersedia ada delapan diagram yang diklasifikasikan berdasarkan tiga jenis diagram sebagai berikut.

Tabel 2.1Klasifikasi Jenis Diagram UML

Structural Diagram	1. Class diagram 2. Component diagram 3. Deployment diagram
Behavioral Diagram	1. Use case diagram 2. Activity diagram 3. Statechart diagram
Interaction Diagram	1. Sequence diagram 2. Collaboration diagram

(Sumber: *Martin Fowler. UML Distilled*)

2.7.3.1 Class Diagram

Class Diagram adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah obyek dan merupakan inti dari pengembangan dan desain berorientasi obyek.

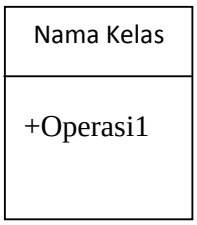
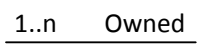
Class menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). Class diagram menggambarkan struktur dan deskripsi class, package dan object beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lainlain. Sebuah Class memiliki tiga area pokok :

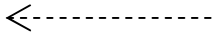
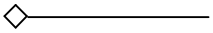
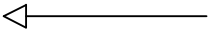
1. Nama, merupakan nama dari sebuah kelas

2. Atribut, merupakan peroperti dari sebuah kelas. Atribut melambangkan batas nilai yang mungkin ada pada obyek dari class
3. Operasi, adalah sesuatu yang bisa dilakukan oleh sebuah class atau yang dapat dilakukan oleh class lain terhadap sebuah class. Atribut dan metoda dapat memiliki salah satu sifat berikut :
    - Private, tidak dapat dipanggil dari luar class yang bersangkutan.
    - Protected, hanya dapat dipanggil oleh class yang bersangkutan dan anak-anak yang mewarisinya.
    - Public, dapat dipanggil oleh siapa saja.
    - Package, hanya dapat dipanggil oleh instance sebuah class pada paket yang sama.

Berikut adalah notasi - notasi yang ada pada class diagram :

Table 2.2 Notasi Class Diagram

Notasi	Keterangan	Simbol
Class	Class adalah blok - blok pembangun pada pemrograman berorientasi objek. Sebuah class digambarkan sebagai sebuah kotak yang terbagi atas 3 bagian. Bagian atas adalah bagian nama dari class. Bagian tengah mendefinisikan property/atribut class. Bagian akhir mendefinisikan method-method dari sebuah class.	
Assosiation	Sebuah asosiasi merupakan sebuah relationship paling umum antara 2 *class*, dan dilambangkan oleh sebuah garis yang menghubungkan antara 2 *class*. Garis ini bisa melambangkan tipe-tipe relationship dan juga dapat menampilkan hukum-hukum multiplisitas pada sebuah relationship (Contoh: One-to-one, one-to-many, many-to-many).	

Composition	Jika sebuah class tidak bisa berdiri sendiri dan harus merupakan bagian dari class yang lain, maka class tersebut memiliki relasi Composition terhadap class tempat dia bergantung tersebut. Sebuah relationship composition digambarkan sebagai garis dengan ujung berbentuk jajaran genjang berisi/solid.	
Dependency	Kadangkala sebuah class enggunakan class yang lain. Hal ini disebut dependency. Umumnya penggunaan dependency digunakan untuk menunjukkan operasi pada suatu class yang menggunakan class yang lain. Sebuah dependency dilambangkan sebagai sebuah panah bertitik-titik.	
Aggregation	Aggregation mengindikasikan keseluruhan bagian relationship dan biasanya disebut sebagai relasi "mempunyai sebuah" atau "bagian dari". Sebuah *aggregation* digambarkan sebagai sebuah garis dengan sebuah jajaran genjang yang tidak berisi/tidak solid.	
Generalization	Sebuah relasi *generalization* sepadan dengan sebuah relasi *inheritance* pada konsep berorientasi objek. Sebuah *generalization* dilambangkan dengan sebuah panah dengan kepala panah yang tidak solid yang mengarah ke kelas "*parent*"-nya/induknya.	

(Sumber : Modul Workshop UML Bab 2)

Class juga dapat merupakan implementasi dari sebuah *interface,* yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class.* Dengan demikian *interface* mendukung resolusi metoda pada saat *runtime.*

Relasi atau *relationship* merupakan keterhubungan antar kelas yang muncul pada saat sebuah kelas berinteraksi dengan kelas-kelas lainnya. Didalam *class diagram,* setiap kelas pasti akan berinteraksi dengan baik satu maupun lebih

kelas. Relasi yang muncul pada setiap keterhubungan antar kelas juga akan memiliki atribut-atribut yang akan lebih menjelaskan definisi dari sebuah relasi yang terjadi.

1. Asosiasi *(Association)*

Para pemodel menggunakan pemahaman asosiasi adalah pada saat beberapa kelas saling terhubung satu sama lain secara konseptual. Asosiasi juga dapat menjadi lebih kompleks pada saat beberapa class terhubung ke satu class.

2. *Constraint* Pada Asosiasi

Kadangkala sebuah asosiasi diantara dua *class* harus mengikuti sebuah aturan dan aturan ini bisa diletakkan dalam sebuah *constraint* pada garis asosiasi dan diletakkan dalam kurung kurawal. Bentuk lain dari tipe *constraint* adalah relasi OR yang ditulis dengan {or} dalam garis putusputus yang menghubungkan dua garis asosiasi. Kondisi OR ini menghadapkan kepada keadaan bahwa sebuah kelas terhubung dengan dua kelas tetapi pada saat mendefinisikan relasinya kelas harus memilih salah satu dari kedua kelas tersebut, sedangkan kondisi untuk memilih keduanya adalah *invalid*.

3. Kelas Asosiasi (*Associations Class*)

Sebuah asosiasi dapat memiliki atribut dan operasi seperti halnya sebuah *class*. Sebuah *association class* sebenarnya diperlukan apabila salah satu dari kelas yang terhubung mempunyai sebuah atau beberapa atribut yang tidak layak dimiliki oleh kelas tersebut, karena secara logis atribut tersebut lebih layak dimiliki oleh asosiasi yang menghubungkan kedua kelas tersebut.

4. Multiplisitas (*Multiplicity*)

Multiplisitas atau kardinalitas adalah jumlah banyaknya obyek sebuah *class* yang berelasi dengan sebuah obyek lain pada *class* lain yang berasosiasi dengan *class* tersebut. Untuk menyatakan multiplisitas anda dapat meletakkannya diatas garis asosiasi berdekatan dengan *class* yang sesuai. Ada banyak multiplisitas yang mungkin untuk dipakai. Tabel berikut menjabarkan multiplisitas yang dapat digunakan.

Tabel 2.3 Nilai Multiplisitas Potensial

Indicator	**Meaning**
0..1	Zero or one
1	One only
0..*	Zero or more
1..*	One to more
N	Only n (where n > 1)
0..n	Zero to n (where n > 1)
1..n	One to n (where n > 1)

(Sumber : Modul Workshop UML Bab 2)

Sebuah *class* (*child class* atau *subclass*) dapat mewarisi atribut-atribut dan operasioperasi dari *class* lainnya (*parent class* atau *super class*) dimana *parent class* bersifat lebih umum daripada *child class.*

Generalisasi pada konsep *Object Oriented* digunakan untuk menjelaskan hubungan kesamaan diantara *class*. Dengan menggunakan generalisasi bisa dibangun struktur logis yang bisa menampilkan derajat kesamaan atau perbedaan diantara *class-class.* Manfaat lain dari struktur hirarkis juga memungkinkan untuk penambahan *subclass* (*child class*) baru tanpa harus merubah struktur yang sudah ada.

Inheritance adalah sebuah mekanisme pengimplementasian generalisasi dan spesialisasi. Aturan inheritance dapat secara umum bisa diklasifikasikan sebagai berikut :

1. *Subclass* selalu mewarisi semua sifat dari *superclass*-nya.
2. Definisi *subclass* selalu mencakup paling tidak satu detil yang tidak diturunkan dari *superclass*-nya.

Inheritance sangat dekat dengan asosiasinya dengan generalisasi. Generalisasi menjelaskan hubungan logis antar elemen-elemen yang mempunyai karakteristik yang sama. Sedangkan *inheritance* menerangkan mekanisme agar bagi pakai (*sharing*) bisa terjadi.

Didalam UML, ada relasi dengan perlakukan khusus yang disebut dengan "bagian dari (*part of*)" yang menangani antar obyek-obyek dimana salah satunya adalah bagian dari yang lain. Dengan kata lain sebuah obyek terdiri atas obyek-obyek yang lain. Hal seperti ini di dalam UML disebut agregasi. Sebuah agregasi adalah kasus khusus dari asosiasi. Agregasi disimbolkan dengan jajaran genjang yang diletakkan pada *class* yang mengandung objek. Kadangkala relasi OR harus digunakan pada agregasi. Untuk memodelkan hal tersebut, sebuah constraint – dengan kalimat atau dalam tanda kurung kurawal pada garis putus-putus harus dibuat untuk menghubungkan dua objek.

Komposit (*composite*) adalah sebuah tipe agregasi yang kuat dimana bagian dari obyek tergantung penuh/secara keseluruhan terhadap objeknya sehingga bila sebuah obyek komposit dibuang maka bagian yang tergantung pada komponen tersebut akan terbuang juga pada saat yang bersamaan. Notasi

komposit sama seperti agregasi hanya saja jajaran genjangnya terisi (solid).

Interface adalah satu set *operation* yang memberikan spesifikasi beberapa aspek dari perilaku dan operasi disuatu *class* ke *class* yang lain. Contohnya, *keyboard* pada komputer sebenarnya merupakan *interface* yang bisa dipakai ulang karena tombol-tombol *keyboard* sebenarnya berasal dari mesin ketik, hanya saja mungkin ada beberapa operasionalisasi tombol-tombol yang berbeda yang sudah ditransfer ke sistem yang lain. Pemodelan *interface* sama dengan pemodelan *class* hanya saja pada *interface* tidak mempunya atribut dan pada penamaannya perlu ditambahkan *stereotype* <<interface>> atau ditambahkan huruf "I" di depan nama interface. Relasi antara *class* dan *interface* disebut *Realization.* Realisasi dituliskan dengan garis putus-putus dengan segitiga yang mengarah ke *interface*.

2.7.3.2 Use Case Diagram

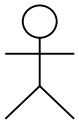
Use case adalah deskripsi fungsi dari sebuah sistem dari perspektif/sudut pandang para pengguna sistem. Use case mendefinisikan "apa" yang dilakukan oleh sistem dan elemen-elemennya, bukan "bagaimana" sistem dan elemen-elemennya saling berinteraksi. Use case bekerja dengan menggunakan "scenario", yaitu deskripsi urutan-urutan langkah yang menerangkan apa yang dilakukan penggunan terhadap sistem maupun sebaliknya. Use case diagram mengidentifikasikan fungsionalitas yang dipunyai oleh sistem (use-case), user yang berinteraksi dengan sistem (actor) dan asosiasi/keterhubungan antara user dengan fungsionalitas sistem.

Fungsi dari diagram use case adalah Menggambarkan Fungsionalitas yang diharapkan dari suatu sistem, dengan menekankan aspek aspek apa yang

dilakukan sistem bukan bagaimana sistem melakukannya. Sebuah use case merepresentasikan sebuah interaksi antara actor dengan sistem.

Berbicara mengenai use case diagram tidak akan terlepas dengan hal yang disebut stereotype. Stereotype adalah sebuah model khusus yang terbatas untuk kondisi tertentu. Untuk menunjukkan stereotype digunakan symbol "<<" diawalnya dan ditutup dengan ">>" diakhirnya. Terdapat 2 stereotype paling sering digunakan dalam use case diagram yaitu <<extend>> dan <<include>>. <<extend>> digunakan untuk menunjukkan bahwa satu use case merupakan tambahan fungsional dari use case yang lain jika kondisi atau syarat tertentu dipenuhi. Sedangkan <<include>> digunakan untuk menggambarkan bahwa suatu use case seluruhnya merupakan fungsionalitas dari use case lainnya.

Table 2.4 Notasi Use Case Diagram

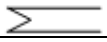
Notasi	Keterangan	Simbol
Actor	Actor adalah pengguna sistem. Actor tidak terbatas hanya manusia saja, jika sebuah sistem berkomunikasi dengan aplikasi lain dan membutuhkan input atau memberikan output, maka aplikasi tersebut juga bisa dianggap sebagai actor.	 Aktor
Use Case	Use case digambarkan sebagai lingkaran elips dengan nama use case dituliskan didalam elips tersebut.	
Association	Asosiasi digunakan untuk menghubungkan actor dengan use case. Asosiasi digambarkan dengan sebuah garis yang menghubungkan antara Actor dengan Use Case.	

(Sumber : Modul Workshop UML Bab 2)

2.7.3.3 Activity Diagram

Activity diagram digunakan untuk mendokumentasikan alur kerja pada sebuah sistem, yang dimulai dari pandangan business level hingga ke operational level. Pada dasarnya, activity diagram merupakan variasi dari statechart diagram. Activity diagram mempunyai peran seperti halnya flowchart, akan tetapi perbedaannya dengan flowchart adalah activity diagram bisa mendukung perilaku parallel sedangkan flowchart tidak bisa. Berikut adalah notasi *activity diagram* :

Table 2.5 Notasi Activity Diagram

Notasi	Keterangan	Simbol
Initial	Titik awal, untuk memulai suatu aktivitas.	
Final	Titik akhir, untuk mengakhiri aktivitas.	
Activity	Activity	
Decision	Pilihan Untuk mengambil Keputusan	
Fork/Join	Digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan peralel menjadi satu.	
Rake	Menunjukkan adanya dekomposisi	
Time	Tanda Waktu	
Send	Tanda pengiriman	
Receive	Tanda penerimaan	
Flow final	Untuk mengkhiri suatu aliran	

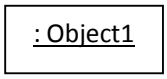
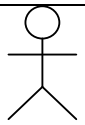
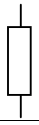
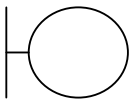
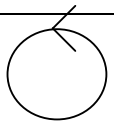
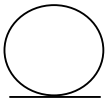
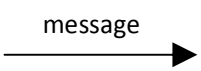
(Sumber : *User Guide Enterprise Architect* 7.0, Modul *Workshop* UML Bab 2)

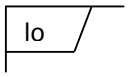
2.7.3.4 Sequence Diagram

Sequence diagram mendokumentasikan komunikasi/interaksi antar kelas-kelas. Diagram ini menunjukkan sejumlah obyek dan *message* (pesan) – yang

diletakkan diantara obyek-obyek didalam use case. Perlu diingat bahwa di dalam diagram ini, kelas-kelas dan aktor-aktor diletakkan dibagian atas diagram dengan urutan dari kiri ke kanan dengan garis lifeline yang diletakkan secara vertikal terhadap kelas dan aktor. Berikut adalah notasi-notasinya :

Table 2.6 Notasi Sequence Diagram

Notasi	Keterangan	Simbol
Object (Partisipan)	*Object* atau biasa juga disebut partisipan merupakan instance dari sebuah class dan dituliskan tersusun secara horizontal. Digambarkan sebagai sebuah class (kotak) dengan nama objek didalamnya yang diawali dengan sebuah titik koma.	
Actor	*Actor* juga dapat berkomunikasi dengan *object*, maka *actor* juga dapat diurutkan sebagai kolom.	
Lifeline	*Lifeline* mengindikasikan keberadaan sebuah object dalam basis waktu. Notasi untuk *Lifeline* adalah garis putus-putus vertikal yang ditarik dari sebuah objek.	
Activation	*Activation* dinotasikan sebagai sebuah kotak segi empat yang digambar pada sebuah *lifeline*. *Activation* mengindikasikan sebuah objek yang akan melakukan sebuah aksi.	
Boundary	Boundary terletak di antara sistem dengan dunia sekelilingnya. Semua form, laporan-laporan, antar muka ke perangkat keras seperti printer atau scanner dan antar muka ke sistem lainnya adalah termasuk dalam kategori.	
Control	Control berhubungan dengan fungsionalitas seperti pemanfaatan sumber daya, pemrosesan terdistribusi, atau penanganan kesalahan.	
Entity	*Entity* digunakan menangani informasi yang mungkin akan disimpan secara permanen. *Entity* bisa juga merupakan sebuah tabel pada struktur basis data.	
Massage	*Message*, digambarkan dengan anak panah horizontal antara *Activation*. Message mengindikasikan komunikasi antara *object-object.*	

Self-Message	Self-message atau panggilan mandiri mengindikasikan komunikasi kembali kedalam sebuah objek itu sendiri.	
Loop	Operator loop adalah *fragmen* yang dapat mengeksekusi berulang kali dan penjaga menunjukkan dasar iterasi.	

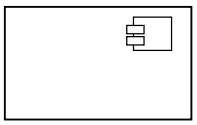
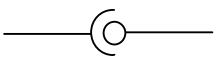
(Sumber : *User Guide Enterprise Architect* 7.0, Modul *Workshop* UML Bab 2)

2.7.3.5 Component Diagram

Komponen perangkat lunak adalah bagian fisik dari sebuah sistem yang menetap di komputer. komponen merupakan implementasi software dari sebuah class. Komponen bisa berupa tabel, file data, file exe, file DLL, dokumen dan lain-lain.

Component diagram mengandung komponen, interface dan relationship. Komponen diagram ini digunakan pada saat anda ingin memecah sistem menjadi komponen-komponen dan ingin menampilkan hubungan-hubungan mereka dengan antarmuka atau pemecahan komponen menjadi struktur yang lebih rendah. Secara umum dapat kita katakan bahwa component diagram kita gunakan untuk menjelaskan kebergantungan antar beragam komponen-komponen software seperti misalnya kebergantungan antara file-file executable dengan file-file sumbernya (source file) dll. Berikut adalah notasi dari *component diagram* :

Tabel 2.7 Notasi Component Diagram

Notasi	Keterangan	Simbol
Component	Sebuah komponen melambangkan sebuah entitas software dalam sebuah sistem. Sebuah komponen dinotasikan sebagai sebuah kotak segiempat dengan dua kotak kecil tambahan yang menempel disebelah kirinya.	
Assembly	Sebuah konektor diantara dua komponen yang salah satunya menyediakan layanan ke komponen lain yang memerlukannya.	

Association	Sebuah *association* digambarkan sebagai sebuah garis yang menghubungkan dua komponen yang mengindikasikan jalur komunikasi antara element-elemen software.	
Depedency	Sebuah *dependency* digunakan untuk menotasikan relasi antara dua komponen. Notasinya adalah tanda panah putus-putus yang diarahkan kepada komponen tempat sebuah komponen itu bergantung.	

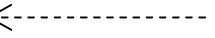
(Sumber : *User Guide Enterprise Architect* 7.0, Modul *Workshop* UML Bab 2)

2.7.3.6 Deployment Diagram

Deployment diagram menunjukkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian software yang berjalan pada bagian-bagian hardware yang digunakan untuk mengimplementasikan sebuah sistem dan keterhubungan antara komponen-komponen hardware tersebut. Deployment diagram dapat digunakan pada bagian-bagian awal proses perancangan sistem untuk mendokumentasikan arsitektur fisik sebuah sistem. Berikut adalah notasi-notasi yang digunakan pada *deployment diagram* :

Tabel 2.8 Notasi Deployment Diagram

Notasi	Keterangan	Simbol
Component	Pada deployment diagram, komponen-komponen yang ada diletakkan didalam node untuk memastikan keberadaan posisi mereka.	
Node	Node menggambarkan bagian-bagian hardware dalam sebuah sistem. Notasi untuk node digambarkan sebagai sebuah kubus 3 dimensi.	Nama
Association	Sebuah association digambarkan sebagai sebuah garis yang menghubungkan dua node yang mengindikasikan jalur komunikasi antara element-elemen hardware.	

Depedency	Sebuah *dependency* digunakan untuk menotasikan relasi antara dua *node*. Notasinya adalah tanda panah putus-putus yang diarahkan kepada *node* tempat sebuah *node* itu bergantung.	

(Sumber : *User Guide Enterprise Architect* 7.0, Modul *Workshop* UML Bab 2)