BAB II

TINJAUAN PUSTAKA

2.1 Pengertian Barang

Menurut Undang-Undang, barang adalah tiap benda dan tiap hak yang dapat menjadi obyek dari hak milik (www.aguschandra.com). Segala sesuatu yang termasuk dalam suatu barang karena hukum perlekatan, begitu pula segala hasilnya, baik hasil alam, maupun hasil usaha kerajinan, selama melekat pada dahan atau akarnya, atau terpaut pada tanah, adalah bagian dari barang itu.

Barang dapat juga didefinisikan sebagai suatu objek atau jasa yang memiliki nilai (aguswibisono.com, 2010). Nilai suatu barang akan ditentukan karena barang itu mempunyai kemampuan untuk dapat memenuhi kebutuhan.

Barang juga dapat diartikan sebagai benda dalam berbagai bentuk dan uraian, yang meliputi bahan baku, barang setengah jadi, barang jadi/peralatan, yang spesifikasinya ditetapkan oleh pengguna barang (www.anneahira.com).

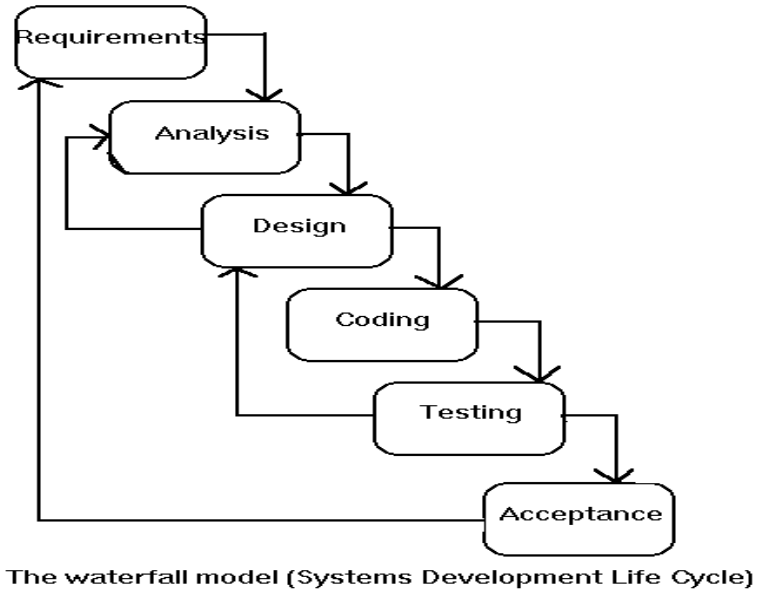
2.2 Pengertian Pengadaan Barang

Secara umum pengadaan barang merupakan fungsi penting dalam suatu perusahaan atau organisasi dan sering melibatkan porsi anggaran yang sangat besar. Fungsi pengadaan sangat dinamis karena mempertemukan kebutuhan pengguna barang / jasa serta ketersediaan dari barang dan jasa tersebut (khalidmustafa.wordpress.com).

Berdasarkan SK No.215.A.SK/3/1/2007, pengadaan barang/jasa adalah usaha atau kegiatan pengadaan barang/jasa yang diperlukan di STT PLN yang meliputi pengadaan barang/jasa pemborong, pemasok, jasa konsultasi dan jasa lainnya.

2.3 SDLC

Systems Development Life Cycle, atau SDLC (Daur hidup pengembangan sistem) adalah proses yang digunakan oleh analis sistem untuk menggembangkan sistem informasi, mulai dari perencanaan, penentuan kebutuhan, perancangan, validasi, sampai pelatihan dan penyerahan kepada konsumen (kelassisteminformasi.blogspot.com,2009).



Gambar 2.1 – Waterfall

SDLC merupakan alur kerja baku yang biasa dipakai oleh perusahaan-perusahaan vendor software dalam mengembangkan *software* aplikasi produksinya. SDLC ini tidak hanya penting untuk proses produksi *software* saja, namun terlebih juga sangat penting untuk proses maintenance *software* itu sendiri,

Tanpa pengarsipan data-data *development* suatu *software*, maka akan sangat menyulitkan perusahaan dalam *maintenance software* tersebut dikemudian hari.

Tahap Perencanaan

- Permasalahan
- Definisi masalah
- Menentukan Tujuan
- Mengidentifikasi kendala sistem

- Studi Kelayakan
- Usulan penelitian sistem
- Menetapkan mekanisme

Tahap Analisis

- Membuat keputusan apabila sistem saat ini mempunyai masalah atau sudah tidak berfungsi secara baik dan hasil analisisnya digunakan sebagai dasar untuk memperbaiki sistem .
- Mengetahui ruang lingkup pekerjaannya yang akan ditanganinya.
- Memahami sistem yang sedang berjalan saat ini.
- Mengidentifikasi masalah dan mencari solusinya.

DESIGN

Mendesain sistem baru yang dapat menyelesaikan masalah-masalah yang dihadapi perusahaan yang diperoleh dari pemilihan alternatif sistem yang terbaik.

- Output design

  Tujuannya adalah memberikan bentuk-bentuk laporan sistem dan dokumennya agar menghasilkan bentuk dari dokumentasi keluaran (*output*).

- Input design

  Bertujuan memberikan bentuk-bentuk masukan didokumen dan dilayar ke sistem informasi agar menghasilkan bentuk dari dokumentasi masukan (*input*).

- File design

  Tujuannya adalah memberikan bentuk-bentuk file-file yang dibutuhkan dalam sistem informasi untuk menghasilkan bentuk dari dokumentasi file.

IMPLEMENTASI

Dalam tahap implementasi memiliki beberapa tujuan, yaitu untuk :

- Melakukan kegiatan spesifikasi rancangan logikal ke dalam kegiatan yang sebenarnya dari sistem informasi yang akan dibangunnya atau dikembangkannya.
- Mengimplementasikan sistem yang baru.
- Menjamin bahwa sistem yang baru dapat berjalan secara optimal.

Keuntungan menggunakan teknik *waterfall*:

- Proses menjadi teratur
- Estimasi proses menjadi lebih baik
- Jadwal menjadi lebih menentu

Kelemahan menggunakan teknik *waterfall:*

- Sifatnya kaku, sehingga susah melakukan perubahan di tengah proses
- Membutuhkan daftar kebutuhan yang lengkap di awal, tapi jarang konsumen bisa memberikan kebutuhan secara lengkap diawal

2.4 UML

UML *(Unified Modeling Language)* adalah metode pemodelan secara visual sebagai sarana untuk merancang dan atau membuat software berorientasi objek (Munawar, 2005). Karena UML ini merupakan bahasa visual untuk pemodelan bahasa berorientasi objek, maka semua elemen dan diagram berbasiskan pada paradigma *object oriented*.

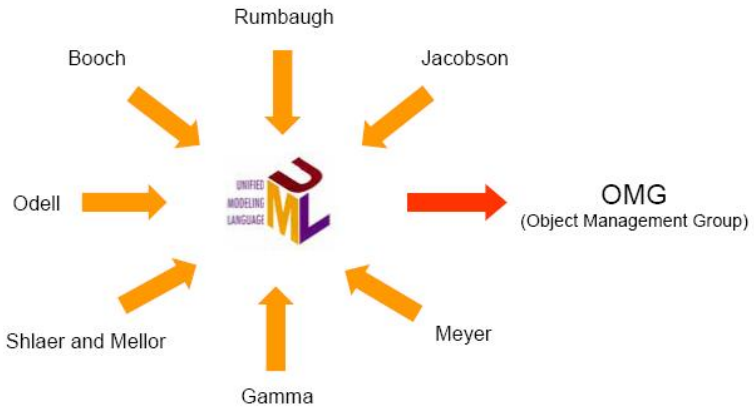
UML adalah salah satu *tool* / model untuk merancang pengembangan software yang berbasis *object oriented*.

UML sendiri juga memberikan standar penulisan sebuah sistem *blue print*, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema database, dan komponen-komponen yang diperlukan dalam sistem software.

Unified Modelling Language (UML) adalah sebuah “bahasa” yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML

menawarkan sebuah standar untuk merancang model sebuah sistem.

Dengan menggunakan UML kita 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 bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C. Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Berikut adalah unsur-unsur pembentuk UML pada **Gambar 2.2**.



Gambar 2.2 – Gambar Unsur-Unsur Pembentuk UML

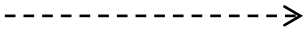
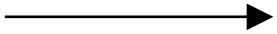
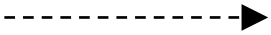
Ada tiga cara dalam memakai UML dalam melakukan pemodelan sistem:

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 system 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.

Ada 4 macam relationship dalam UML, berikut ditunjukkan pada **Tabel 2.1** di bawah ini:

Tabel 2.1 – Tabel Relasi pada UML

Dependency	
Association	
Generalization	
Realization	

2.4.1 Diagram-Diagram UML

UML 2 terdiri dari 13 jenis diagram resmi seperti tertulis dalam **Tabel 2.2**. Meskipun jenis-jenis diagram ini merupakan cara orang-orang memperlakukan UML dan cara mengorganisasikan buku ini, para perancang UML tidak memandang diagram sebagai bagian yang sentral. Dan hasilnya, jenis-jenis diagram sebagai bagian yang sentral. Acapkali secara legal dapat menggunakan elemen-elemen dari satu jenis diagram untuk diagram yang lain. Standard UML menunjukkan bahwa elemen-elemen tertetu hanya diambil dari jenis diagram tertentu, tetapi ini bukanlah hal yang dianjurkan.

Tabel 2.2 – Tabel Diagram-Diagram pada UML

Diagram	Kegunaan	Turunan
Activity	Behavior procedural dan paralel	Di UML 1
Class	Class, fitur, dan hubungan-hubungan	Di UML 1
Communication	Interaksi pada objek; penekanan pada jalur	Diagram kolaborasi UML 1
Component	Struktur dan koneksi komponen	Di UML 1
Composite structure	Dekomposisi runtime sebuah class	Baru di UML 2
Deployment	Pemindahan artifak ke node	Di UML 1
Interaction overview	Campuran sequence dan activity diagram	Baru di UML 2
Object	Contoh konfigurasi dari contoh-contoh	Tidak resmi di UML 1
Package	Struktur hirarki compile-time	
Sequence	Interaksi antar objek; penekanan pada sequence	Di UML 1
State machine	Bagaimana even mengubah objek selama aktif	Di UML 1
Timing	Interaksi antar objek; penekanan pada timing	Baru di UML 2
Use Case	Bagaimana pengguna	Di UML 1

	berinteraksi dengan sebuah sistem	

Diagram ini digunakan untuk :

- Mengkomunikasikan ide
- Melahirkan ide-ide baru dan peluang-peluang baru
- Menguji ide dan membuat prediksi
- Memahami struktur dan relasi-relasinya

2.4.2 *Use Case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem (Sholiq, 2006). Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan system untuk melakukan pekerjaan-pekerjaan tertentu.

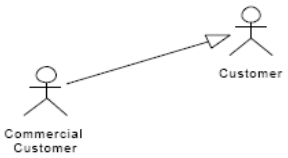
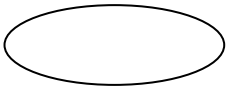
Use case diagram dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan

merancang *test case* untuk semua *feature* yang ada pada sistem.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*.

Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupaka spesialisasi dari yang lain. Komponen-komponen yang ada dalam use case antara lain:

Tabel 2.3 – Tabel *Actor*

Gambar	Keterangan
	*Actor*
	*Use Case*

1. Actor

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

Ada 3 tipe aktor : pengguna sistem, sistem lain yang berhubungan dengan sistem yang sedang dibangun, dan waktu. Tipe pertama, aktor secara fisik, atau seorang pengguna. Ini adalah gambaran aktor secara umum, dan selalu ada pada setiap sistem.

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

- Yang berkepentingan terhadap sistem dimana adanya arus informasi baik yang diterimanya maupun yang dia inputkan ke sistem.

- Orang ataupun pihak yang akan mengelola sistem tersebut.
- *External resource* yang digunakan oleh sistem.
- Sistem lain yang berinteraksi dengan sistem yang akan dibuat.

2. Use Case

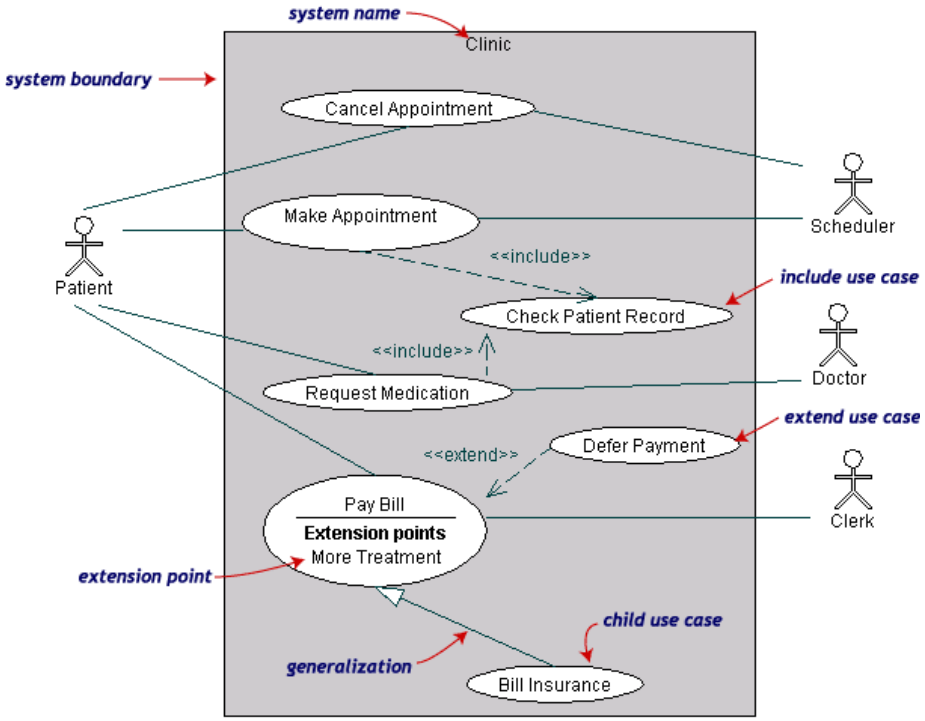
Use case adalah gambaran fungsionalitas dari suatu sistem, sehingga *customer* atau pengguna sistem paham dan mengerti mengenai kegunaan sistem yang akan dibangun.

Use case merupakan bagian tingkat tinggi dari fungsionalitas yang disediakan oleh sistem. Dengan kata lain, *use case* menggambarkan bagaimana seseorang menggunakan sistem.

Cara menentukan Use Case dalam suatu sistem:

- Pola perilaku perangkat lunak aplikasi.
- Gambaran tugas dari sebuah *actor*.
- Sistem atau “benda” yang memberikan sesuatu yang bernilai kepada *actor*.
- Apa yang dikerjakan oleh suatu perangkat lunak (*bukan bagaimana cara mengerjakannya).

Berikut adalah contoh dari *Use Case Diagram* :



Gambar 2.3 – *Use Case Diagram*

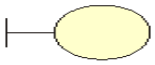
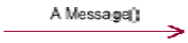
2.4.3 *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* 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 respons dari sebuah *event* 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.

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah *scenario*. Diagram ini menunjukkan sejumlah contoh obyek dan message (pesan) yang diletakkan di antara obyek-obyek ini di dalam *Use Case*.

Tabel 2.4 – Tabel Simbol *Sequence Diagram*

a.	An Actor 	Menggambarkan orang yang sedang berinteraksi dengan sistem
b.	Entity Class 	Menggambarkan hubungan kegiatan yang akan dilakukan
c.	Boundary Class 	Mengganbarkan sebuah penggambaran dari form
d.	Control Class 	Menggambarkan penghubung antara boundary dengan tabel
e.	A focus Of Control & A life line 	Menggambarkan tempat mulai dan berakhirnya sebuah massage
f.	A massage 	Menggambarkan Pengiriman Pesan

2.4.4 *Activity Diagram*

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-

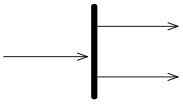
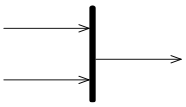
masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

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 (dan interaksi antar subsistem) 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.

Sama seperti *state*, standar UML menggunakan segiempat 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. *Activity diagram* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

Tabel 2.5 – Tabel Simbol *Activity Diagram*

Simbol	Keterangan
	Start Point
	End Point
	Activities
	Fork (Percabangan)
	Join (Penggabungan)
	Decision
Swimlane	Sebuah cara untuk mengelompokkan activity berdasarkan Actor (mengelompokkan activity dalam sebuah urutan yang sama)

2.4.5 *Class Diagram*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek (Nugroho, 2010). *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 objek beserta hubungan satu sama lain seperti

containment, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok :

- Nama (dan stereotype)
- Atribut
- Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

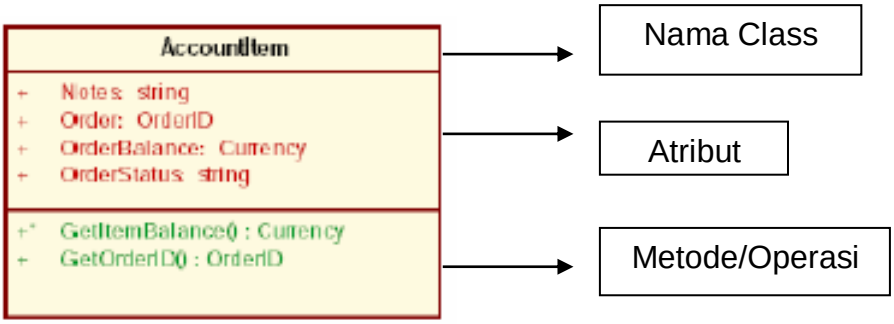
- *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan
- *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
- *Public*, dapat dipanggil oleh siapa saja

Hubungan Antar *Class* :

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian (“terdiri atas..”).
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas

baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.

4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.



Gambar 2.4 – *Class Diagram*

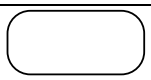
2.4.6 *Statechart Diagram*

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima (Nugroho,2005). Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).

Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki

kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring. Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.

Tabel 2.6 – Tabel *Statechart Diagram*

Gambar	Keterangan
	Status
	Mulai
	Henti

2.4.7 *Component Diagram*

Component adalah sebuah *code module* (kode-kode module). *Diagram Component* merupakan fisik sebenarnya dari *Diagram Class*.

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan *(dependency)* diantaranya. Komponen piranti lunak adalah modul berisi code, baik berisi *source*

code maupun *binary code,* baik *library* maupun *executable,* baik yang muncul pada *compile time, link time maupun run time.* Umumnya komponen terbentuk dari beberapa class dan atau package, tapi dapat juga berupa *interface,* yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lainnya.

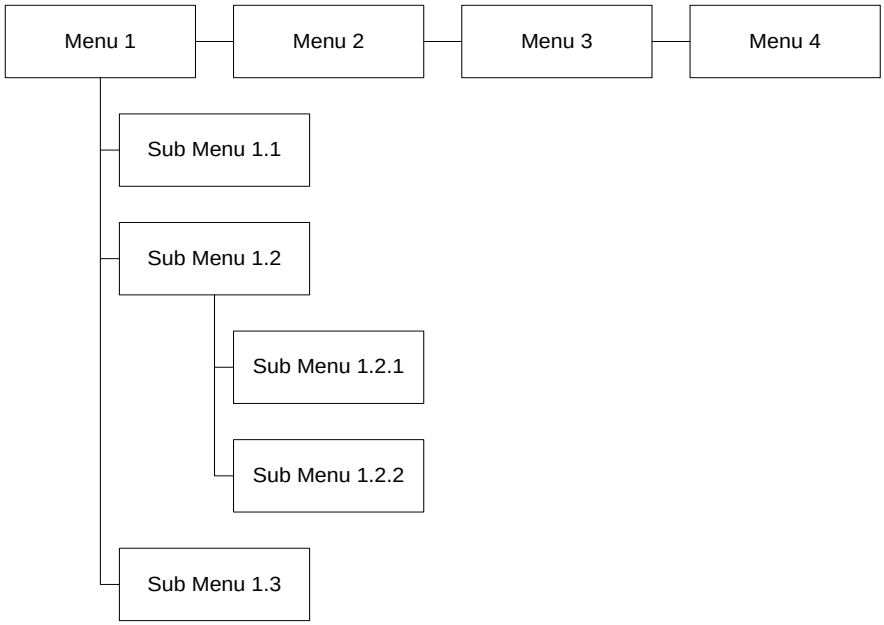
2.4.8 *Deployment Diagram*

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, dimana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal. Sebuah node adalah *server, workstation* atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya.

2.5 Hirarki Menu

Dalam hirarki menu menjelaskan tentang jalannya alur suatu menu dan hubungan antar menu.

Berikut ini pada **Gambar 2.5** adalah contoh dari hirarki menu:



Gambar 2.5 - Hirarki Menu

2.6 Konsep Aplikasi Berbasis Web PHP

Pada prinsipnya aplikasi berbasis web PHP bekerja dengan cara menampilkan file-file yang telah diproses yang berasal dari server web pada program client khusus, yaitu browser web (Sukarno, 2006). Program browser pada client mengirimkan permintaan (request) kepada server web, kemudian server mengirim request file PHP ke PHP parser untuk diproses yang kemudian file HTML hasil parser dari file PHP, dikirirm kembali ke server dan dilanjutkan dikirim ke browser client dalam bentuk HTML sehingga file dapat ditampilkan secara visual kepada pengguna di layar komputer.

PHP merupakan script untuk pemograman webserver-side, yaitu script yang membuat dokumen HTML secara *on the fly.*

Dokumen HTML yang dihasilkan dari suatu aplikasi bukan dokumen HTML yang dibuat dengan menggunakan editor teks atau editor HTML.

PHP banyak digunakan karena :

- PHP dapat dijalankan pada platform yang berbeda-beda (Windows, Linux, Unix, etc.) .
- PHP merupakan web scripting open source .
- PHP mudah dipelajari.

2.6.1 Kelebihan PHP

Di antara maraknya pemrograman server web saat ini, adalah ASP yang berkembang menjadi ASP.NET, JSP, CFML, dan PHP. Jika dibandingkan di antara 3 terbesar pemrograman server web di atas, terdapat kelebihan dari PHP itu sendiri, yaitu :

- PHP merupakan bahasa script yang tidak melakukan sebuah kompilasi dalam penggunaannya. Tidak seperti halnya bahasa pemrograman aplikasi seperti Visual Basic dan sebagainya.
- PHP dapat berjalan pada web server yang dirilis oleh Microsoft, seperti IIS dan PWS juga pada Apache yang bersifat Open Source.
- Karena sifatnya yang Open Source, maka perubahan dan perkembangan interpreter pada PHP lebih cepat dan mudah,

karena banyak milis-milis dan developer yang siap membantu pengembangannya.

- Jika dilihat dari segi pemahaman, PHP memiliki referensi yang begitu banyak sehingga sangat mudah dapat dipahami.
- PHP dapat berjalan pada 3 *operating system*, yaitu : Linux, Unux, dan Windows, dan juga dapat dijalankan secara *runtime* pada suatu console.

2.7 HTML

HTML atau Hypertext Markup Language merupakan protokol yang digunakan untuk mentransfer data atau document dari web server ke browser (Internet Explorer, Netscape Navigator, NeoPlanet, dll) (Paranginangin, 2006).

HTML adalah bahasa yang sangat tepat dipakai untuk menampilkan informasi pada halaman Web karena HTML menampilkan informasi dalam bentuk hypertext dan juga mendukung sekumpulan perintah yang dapat digunakan untuk mengatur tampilnya informasi tersebut.

Sesuai dengan namanya, bahasa ini menggunakan tanda (markup) untuk menandai perintah-perintahnya. Saat ini banyak sekali aplikasi yang dapat digunakan untuk membuat Web Page secara mudah, seperti Microsoft FrontPage, Adobe Golive dan lainnya. Namun demikian untuk seorang Web Developer harus memiliki kemampuan dasar menguasi perintah HTML.

Untuk dapat menggunakan HTML, dibutuhkan beberapa hal, diantaranya adalah:

a. Anda memerlukan teks editor untuk mengetikkan HTML Anda. Klik Start | Program kemudian Accessories lalu pilih Notepad
b. Anda membutuhkan sebuah web browser untuk mempublikasikan HTML Anda. Anda dapat menggunakan Microsoft Internet Explorer.
c. Anda membutuhkan tempat penyimpanan. Gunakan hard disk, floppy disk, atau web server. Anda tidak harus bekerja online dengan internet, Anda dapat menulis HTML kemudian menggunakan web browser secara offline di komputer.

2.7.1 Struktur dasar HTML

HTML (Hypert Text Markup Language) merupakan bahasa pemrograman yang digunakan dalam pembuatan halaman web. Dalam penggunaannya sebagian besar kode HTML tersebut harus terletak di antara tag kontainer. Yaitu diawali dengan <namatag> dan diakhiri dengan </namatag> (terdapat tanda "/").

Sebuah halaman web minimal mempunyai 3 buah tag, yaitu:

<HTML> Sebagai tanda awal dokumen HTML.

<HEAD> Sebagai informasi page header. Di dalam tag ini kita bisa meletakkan tag-tag TITLE, BASE, ISINDEX, LINK, SCRIPT, STYLE & META.

<TITLE> Sebagai titel atau judul halaman. Kalimat yang terletak di dalam tag ini akan muncul pada bagian paling atas browser Anda (pada title bar).

2.7.2 Pengaturan Teks

Untuk mendapatkan halaman web yang baik maka harus melakukan pengaturan terhadap teks seperti memilih jenis dan ukuran huruf, perataan, dll.

2.8 MySQL

2.8.1 Mengenal MySQL

MySQL adalah sebuah sistem manajemen database yang saling berhubungan (Sugianto, 2003). MySQL merupakan sebuah perangkat lunak sistem manajemen basis data dengan menggunakan standard SQL atau RDBMS (*Relational Database Manajemen System)* yang *multi-thread, multi-user* dengan sekitar 10 juta instalasi di seluruh dunia.

2.8.2 Kelebihan My SQL

Sebagai software database dengan konsep database modern, MySQL memiliki banyak kelebihan, yaitu sebagai berikut :

- *Protability*

  MySQL dapat digunakan dengan stabil tanpa kendala, berarti pada berbagai sistem operasi diantaranya seperti Windows, Linux, Mac OS X Server, Solaris, Amiga HP-UX dan masih banyak lagi.

- *Open source* MySQL didistribusikan secara *open source* di bawah lisensi GPL, sehingga dapat memperoleh menggunakannya secara cuma-cuma tanpa dipungut biaya sepeserpun.

- *Multiuser*

  MySQL dapat digunakan untuk menangani beberapa user dalam waktu yang bersamaan tanpa mengalami masalah atau konflik. Hal ini akan memungkinkan sebuah database server MySQL dapat diakses client secara bersamaan dalam waktu yang bersamaan pula.

- *Performance Tuning*

  MySQL memiliki kecepatan yang cukup menakjubkan dalam menangani query sederhana, serta mampu memproses lebih banyak SQL persatuan waktu.

- *Column Types*

  MySQL didukung tipe kolom (tipe data) yang sangat kompleks.

- *Command dan Functions*

  MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah SELECT dan WHERE dalam query.

- *Scalability dan Limits*

  Dalam hal batas kemampuan, MySQL terbukti mampu menangani database dalam skala yang besar dengan jumlah record lebih dari 50 juta dan 60 ribu tabel serta 5 miliar baris. Selain itu batas indeks yang dapat ditampung mencapai 32 indeks pada setiap tabelnya.

- *Interface*

  Sama halnya dengan *software* database lainnya, MySQL memiliki *interface* (antarmuka) terhadap berbagai aplikasi

dan bahasa pemrograman dengan menggunakan fungsi API (*Aplication Programming Interface*).

- Struktur tabel

  Struktur tabel MySQL cukup baik, serta cukup fleksibel. Misalnya ketika menangani Alter Table, dibandingkan database lainnya semacam ProgresSQL ataupun Oracle.

2.9 Appserv

AppServ adalah paket installer untuk Windows yang membundel program opensource yang terdiri dari :

- Apache Web Server
- PHP Script Language
- MySQL Database
- phpMyAdmin Database Manager

AppServ merupakan sebauh Software yang berfungsi sebagai Virtual Web-hosting atau Virtual Server yang mendukung sebagai Aplikasi untuk di jadikan Web Server.

2.10 Javascript

Javascript adalah bahasa skrip yang ditempelkan pada kode HTML dan diproses di sisi klien. Dengan adanya bahasa ini, kemampuan dokumen HTML menjadi semakin luas. Sebagai contoh, dengan menggunakan JavaScript dimungkinkan untuk memvalidasi masukan pada formulir sebelum formulir dikirimkan ke *server*. Javascript bukanlah bahasa Java dan merupakan dua bahasa yang berbeda. Javascript diinterpretasikan oleh klien (kodenya bisa dilihat pada sisi klien), sedangkan kode Java dikompilasi oleh pemrogram dan hasil kompilasinyalah yang dijalankan oleh klien.

2.10.1 Struktur JavaScript

Struktur dari JavaScript adalah sbb :

<SCRIPT LANGUAGE = ”JavaScript”>

<!- -

Penulisan kode javascript

// - - >

</SCRIPT>

Keterangan :

Kode <!- - // - - > umumnya disertakan dengan tujuan agar sekiranya browser tidak mengenali JavaScript maka browser akan memperlakukannya sebagai komentar sehingga tidak ditampilkan pada jendela browser.

2.10.2 JavaScript Pada Obyek Properti

Properti adalah atribut dari sebuah objek. Contoh, objek mobil punya properti warna mobil. Penulisan :

Nama_objek.nama_properti = nilai

window.defaultStatus = ”Selamat Belajar JavaScript”;

Metode adalah suatu kumpulan kode yang digunakan untuk melakukan sesuatu tindakan terhadap objek. Penulisan :

Nama_objek.nama_metode(parameter)

document.write (”Hallo”)

2.10.3 Letak JavaScript dalam HTML

Script Javascript dalam dokumen HTML dapat diletakkan pada :

- Bagian Head
- Bagian Body (jarang digunakan).