# BAB II

# TINJAUAN PUSTAKA

## II.1. *Short Messaging Services* (SMS)

SMS adalah bagian (*fitur*) dari GSM, dan merupakan teknologi yang memungkinkan pengiriman dan penerimaan pesan (*message*) dalam bentuk teks antar *mobile phone*. Data yang dapat dibawa oleh SMS sangat terbatas. Satu pesan SMS dapat memuat :

1. Maksimum 160 karakter jika menggunakan encoding karakter 7-bit (biasanya digunakan untuk encoding huruf Latin).
2. Maksimum 140 karakter jika menggunakan encoding karakter 8-bit (biasanya digunakan untuk mengirimkan ringtone dan image – smart messaging).
3. Maksimum 70 karakter jika menggunakan encoding karakter 16-bit Unicode (untuk SMS yang memuat huruf non-Latin seperti China, Jepang, Arab).

Selaing teks, SMS juga dapat memuat data binary, misalnya *logo, ringtone, business card* (vCard) dan konfigurasi *Wiresless Applicaton Protocol* (WAP) [1].

Dalam teknologi SMS terdapat istilah SMS *Center* (SMSC). SMSC bertugas untuk menangani SMS. Saat suatu SMS dikirim dari *mobile phone*, SMS tersebut akan diterima oleh SMSC, kemudian SMSC ini akan melakukan *forwarding* ke *mobile phone* tujuan jika *mobile phone* tujuan sedang aktif. Jika *mobile phone* tujuan sedang tidak aktif, maka SMSC akan menyimpan (*store*) SMS tersebut, dan akan mengirimkannya nanti jika *mobile phone* tujuan menjadi aktif . Jika *mobile phone* tujuan tidak aktif dalam waktu tertentu, maka SMS

tersebut akan dihapus dari SMSC. Waktu ini disebut *validity period*. Umumnya suatu operator mempunyai SMSC-nya tersendiri, dan alamat/nomor dari SMSC tersebut telah ada pada SIM *card* dari operator tersebut. Sebagai contoh, SMSC dari indosat M3 adalah +6285000000. Selain *validity period*, hal-hal lain yang terdapat pada teknologi SMS untuk memberikan informasi mengenai pengiriman dan penerimaan adalah *Message Status Report*, Message *Submission Report*, dan *Message Delivery Report* [1]. (Jurnal Informatika, Cahyo Rossy W, Wiranto Herry Utomo,Theophillus Wellen ,Vol.2 No.2.Desember 2006: 155 – 166).

## II.2. Cara Kerja *Short Message Service* (SMS)

Pesan SMS dikirim dari suatu *Mobile Station* (MS) pengirim ke MS penerima melalui *SMS Center* (SMSC), yang bertindak sebagai sistem simpan dan terusan (*store and forward*). Dengan sistem ini MS pengirim mengirim pesan (*store*) ke SMSC, dan kemudian oleh SMSC, pesan ini diteruskan (*forward*) ke MS tujuan /penerima. Keuntungan mekanisme ini adalah, MS penerima tidak perlu berada dalam kondisis *online* ketika ada pengirim yang bermaksud mengirim SMS kepadanya, karena pesan akan disimpan sementara di SMSC, dan akan diteruskan oleh SMSC ketika penerima berada dalam kondisi *online* di lain waktu[4]. Mekanisme ini ditunjukan pada gambar II.1


**Gambar II.1 *Mekanisme Store and Forward* pemgiriman pesan SMS**

(Sumber : Seminar Nasional Aplikasi Teknologi Informasi, Hendrik, 2007)

Sistem pengingat adalah program sistem komunikasi dua arah yaitu SMS keluar dan SMS masuk dengan memanfaatkan komponem *Gammu* sebagai *software* komunikasi antara komputer dengan *handphone*.

## II.3. SMS *Gateway*

SMS *gateway* memungkinkan kita mengirimkan dan menerima SMS dari/ke perangkat bergerak/telepon seluler ke perangkat lain selain telepon seluler. Adapun aplikasi SMS *gateway* digunakan untuk menangani atau mengelola pesan SMS dari pengguna dengan aturan tertentu sehingga dapat mengirim/menerima pesan SMS dari/ke berbagai media (missal: *email* ke SMS atau sebaliknya, SMS ke *Skype* atau sebaliknya, dan lain-lain. Saat ini telah banyak aplikasi SMS *gateway* yang tersedia baik bersifat komersial maupun *free* seperti SMS *Tools*, *Kannel* (http://www.kannel.org), dan lain-lain.(Seminar Nasional Aplikasi Teknologi informasi, Hendrik, 16 Juni 2007: A-38).

## II.4. *Database*

*Database* sering didefinisikan sebagai kumpulan data yang terkait. Secara teknis, yang berada dalam sebuah *database* adalah sekumpulan tabel atau objek lain (*indeks, view*, dan lain-lain). Tujuan utama pembuatan *database* adalah untuk memudahkan dalam mengakses data. Data dapat ditambahkan, diubah, dihapus, atau dibaca dengan relatif mudah dan cepat.

Sebuah tabel (atau kadang disebut relasi) berisi sejumlah baris dan kolom menyatakan sebuah data. Saat ini tersedia banyak perangkat lunak yang ditujukan

untuk mengelola *database*. Perangkat lunak seperti itu biasa dinamakan DBMS (*database management system*). *Access, Ms SQL Server*, dan *MySQL* merupakan kelas *database server*, yaitu jenis yang secara aktif memantau permintaan akses terhadap data. Dalam hal ini, *database server* akan segera menanggapi permintaan data. Adapun yang bukan termasuk *database server* adalah *Access*.

(Abdul Kadir, 2009:14-15)

Data dalam sebuah *database* disusun berdasarkan sistem *hierarki* yang *unik*, yaitu:

1. *Database*, merupakan kumpulan *file* yang saling terkait satu sama lain, misalnya *file* data induk karyawan, *file* jabatan, *file* penggajian dan lain sebagainya. Kumpuan *file* yang tidak saling terkait satu sama lain tidak dapat disebut *database*, misalnya *file* data induk karyawan, *file* tamu undangan perkawinan, *file* barang *retail* pasar swalayan.
2. *File*, yaitu kumpulan dari *record* yang saling terkait dan memiliki format *field* yang sama dan sejenis.
3. *Record*, yaitu kumpulan *field* yang menggambarkan suatu unit data individu tertentu.
4. Field, yaitu atribut dari *record* yang menunjukkan suatu item dari data seperti nama, alamat, dan lain sebagainya.
5. *Byte*, yaitu atribut dari *field* yang berupa huruf yang membentuk nilai dari sebuah *field*. Huruf tersebut dapat berupa numerik maupun abjad atau karakter khusus.

6. *Bit*, yaitu bagian terkecil dari data secara keseluruhan, yaitu berupa karakter ASCII nol atau satu yang merupakan komponen pembentuk *byte*.
   (Budi Sutedjo Dharma Oetomo, S.Kom; 2006:102)

## II.5. Defenisi Perangkat Lunak

Berikut adalah perangkat lunak yang akan digunakan untuk membuat sistem informasi yang akan dibangun.

### II.5.1. Gammu

*Gammu* adalah nama sebuah *project* yang ditujukan untuk membangun aplikasi, *script* dan *drivers* yang dapat digunakan untuk semua fungsi yang memungkinkan pada telepon seluler atau alat sejenisnya. Sekarang gammu telah menyediakan *codebase* yang stabil dan mapan untuk berbagai macam model telepon yang tersedia di pasaran dibandingkan dengan project sejenis. Gammu merupakan *project* yang berlisensi GNU GPL 2 sehingga menjamin kebebasan menggunakan *tool* ini tanpa perlu takut dengan masaah legalitas dan biaya yang mahal yang harus dikeluarkan. Gammu mendukung berbagai macam model telepon seluler dengan berbagai jenis koneksi dan *type*. (www.gammu.org).

Gammu merupakan *software* sms *gateway* yang cukup bagus dan terkenal. Selain mudah penggunaannya, perangkat *modem* gsm yang *support* cukup banyak mulai dari nokia, siemen dan Sonny ericsson. Selain itu perangkat lain yang lebih cocok untuk dijadikan sms *gateway* dengan *software* gammu seperti *modem* gsm itegno, wavecom dan lain-lain. Gammu bahkan sudah menyediakan *service online*

untuk proses *update* data sms ke *databas*e. *Database* yang di *support* Gammu adalah Mysql.

Sebenarnya untuk membangun SMS *gateway* banyak sekali *software* yang menyediakan layanan tersebut seperti GAMPS SMS, MitraSMS, Gnokii dan lain-lain sebagainya.

Kelebihan Gammu:

Kelebihan Gammu dari *tool* sms *gateway* lainnya adalah :

1. Gammu bisa di jalankan di *Windows* maupun *Linux*.

2. Banyak *device* yang *kompatibel* oleh gammu.

3. Gammu menggunakan *database* MySql.

4. Baik kabel data *USB* maupun *SERIAL*, semuanya *kompatibe*l di Gammu.

**II.5.2. PHP**

PHP singkatan dari PHP : *Hypertext Preprocessor* yaitu bahasa pemograman *web server-side* yang bersifat *open source*. PHP merupakan *script* yang terintegrasi dengan HTML dan berada pada *server* (*server side HTML embedded scripting*). PHP adalah *script* yang digunakan untuk membuat halaman *website* yang dinamis. Dinamis berarti halaman yang akan ditampilkan dibuat saat halaman itu diminta oleh *client*. Mekanisme ini menyebabkan informasi yang diterima *client* selalu yang terbaru*/up to date*. Semua *script* PHP dieksekusi pada *server* di mana *script* tersebut dijalankan.

Sebenarnya saat kita berinternet menggunakan *browser* seperti Mozilla, Internet Explorer, Opera, d an Safari, halaman yang muncul di depan layar komputer kita adalah halaman yang diperoleh dari proses pemanggilan dengan menuliskan alamat sesuai nama *file* yang terdapat pada *web server*. *Web server* adalah aplikasi yang berfungsi untuk melayani permintaan pemanggilan alamat dari pengguna melalui *web browser*, di mana *web server* mengirimkan kembali informasi yang diminta tersebut melalui HTTP (*HiperText Transfer Protocol*) untuk ditampilkan ke layar monitor komputer kita. Agar kita dapat mengubah isi dari *website* yang dibuat, kita membutuhkan program PHP. *Script-script* PHP tersebut yang berfungsi membuat halaman *website* menjadi dinamis. Dinamis artinya pengunjung *web* dapat memberikan komentar saran/masukan pada *website* kita. *Website* yang kita buat menjadi lebih hidup karena ada komunikasi antara pengunjung dan kita sebagai *webmaster*nya.(Anhar, 2010 :3-4)

### II.5.3. MySQL

MySQL (*My Structure Query Language*) adalah salah satu *DataBase Management System* (DBMS) dari sekian banyaknya DBMS seperti Oracle, MS SQL, Postagre SQL, dan lainnya. MySQL berfungsi untuk mengolah *database* menggunakan bahasa SQL. MySQL bersifat *open source* sehingga kita biasa menggunakannya secara gratis. Pemograman PHP juga sangat mendukung/*support* dengan *database* MySQL. Sebuah website yang dinamis membutuhkan tempat penyimpanan data agar pengunjung dapat memberikan komentar, saran, dan masukan atas *website* yang dibuat. Tempat penyimpanan

data berupa informasi dalam sebuah tabel disebut dengan *database*. Program yang digunakan untuk mengolah dan mengelola *database* adalah MySQL yang memiliki sekumpulan prosedur dan struktur sedemikian rupa sehingga mempermudah dalam penyimpanan , mengatur, dan menampilkan data. Agar memudahkan kita dalam mempelajari dasar pemrograman MySQL.(Anhar, 2010 :45)

### II.5.4. Adobe Dreamweaver CS3

Dreamweaver adalah sebuah HTML editor profesional untuk mendesain *web* secara *visual* dan mengelola situs *web* maupun halaman *web*. Saat ini terdapat *software* dari kelompok adobe yang belakangan banyak digunakan untuk mendesain suatu situs *Web. Versi* terbaru dari Adobe Dreamweaver saat ini adalah Dreamweaver CS3. Pada Dreamweaver CS3 terdapat beberapa kemampuan bukan hanya sebagai *software* untuk mendesain *web* saja tetapi juga untuk menyunting kode serta pembuatan aplikasi *Web* dengan menggunakan berbagai bahasa pemrograman *Web*, antara lain: JPS, PHP, ASP dan *ColdFusion.*

Dreamweaver merupakan *software* utama yang digunakan oleh *Web Desaigner* maupun *Web Programmer* dalam mengembangkan suatu situs *Web*. Hal ini disebabkan ruang kerja, fasilitas dan kemampuan Dreamweaver yang mampu meningkatkan produktivitas dan efektivitas dalam desain maupun membangun suatu situs *Web*.

1. Fasilitas Dalam Dreamweaver CS3

Dreamweaver CS3 memiliki peningkatan kemampuan *toolbar*, dimana Dreamweaver CS3 dapat digunakan untuk memodifikasi tampilan *toolbar* atau menambahkan fungsi baru. Selain *user interface* baru, Dreamweaver CS3 memiliki kemampuan untuk menyunting kode dengan lebih baik. Dreamweaver CS3 juga dapat melakukan print kode pada jendela *Code View*, selain itu juga memiliki fasilitas *Code Hints* yang membantu dalam urusan *tag-tag*, serta *Tag Inspector* yang sangat berguna dalam menangani *tag-tag* HTML.

2. Menjalankan Dreamweaver CS3

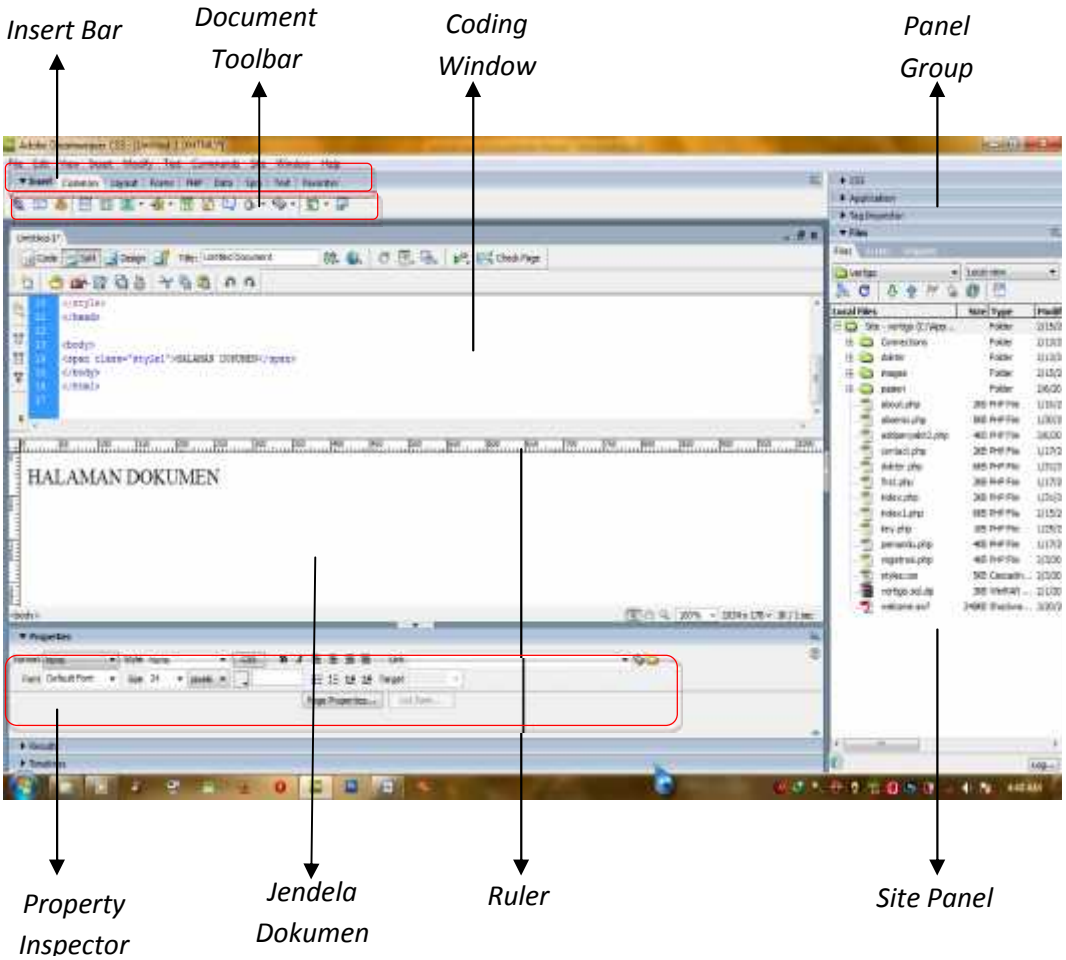
Langkah untuk memulai Adobe Dreamweaver CS3 adalah : *klik* tombol ***Start,*** pilih ***All Programs,*** pilih **Dreamweaver CS3**. Setelah itu tampilan awal dari Dreamweaver CS3 akan terbuka seperti gambar II.2 berikut ini :


**Gambar II.2 Tampilan awal halaman Adobe Dreamweaver CS3**

(Sumber : Andi : 2008)

3. Ruang Kerja Dreamweaver CS3

Ruang kerja Dreamweaver CS3 memiliki komponen-komponen yang memberikan fasilitas dan ruang untuk menuangkan kreasi anda saat bekerja, seperti yang terlihat pada gambar II.3. Komponen-komponen yang disediakan oleh ruang kerja Dreamweaver CS3 antara lain *Insert Bar*, *Document Toolbar*, Jendela Dokumenn *Panel Group*, *Tag Selector*, *Property Inspector*,dan *Site Panel.*




**Gambar II.3 Tampilan ruang kerja tipe Coder**

(Sumber : Andi : 2008)

Penjelasan dari komponen-komponen yang terdapat di dalam ruang kerja Dreamweaver CS3 adalah :

a. ***Insert Bar***, berisi tombol-tombol untuk meyisipkan berbagai macam objek seperti: *image*, tabel dan *layer* ke dalam dokumen.

b. ***Document Toolbar***, berisi tombol-tombol dan menu *pop-up* yang menyediakan tampilan berbeda dari jendela dokumen.

c. ***Coding Window***, berisi kode-kode HTML dan tempat untuk menuliskan kode-kode pemrograman, misalnya PHP atau ASP.

d. ***Panel Group***, adalah kumpulan panel yang saling berkaitan satu sama lainnya yang dikelompokkan dibawah satu judul.

e. ***Property Inspector***, digunakan untuk melihat dan mengubah berbagai *properti* objek atau teks.

f. **Jendela Dokumen**, berfungsi untuk menampilkan dokumen dimana anda sekarang bekerja.

g. ***Ruler***, mempermudah ukuran dalam mendesain halaman *Web.*

h. ***Site Panel***, digunakan untuk mengatur *file-file* dan *folder-folder* yang membentuk situs *Web* anda.(Madcoms,2008:3-6)

## II.6. UML (*Unified Modelling Language*)

UML (*Unified Modelling Language*) adalah salah satu alat bantu yang sangat handal di dunia perkembangan sistem yang berorientasi objek. Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi perkembangan sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (sharing) dan mengkomunikasikan rancangan dengan baik (Munawar ; 2005 : 17).

UML merupakan kesatuan bahasa pemodelan yang dikembangkan oleh Booch, *Object Modeling Technique* (OMT) dan *Object Oriented Engineering* (OOSE). Metode Booch dari Grady Booch sangat terkenal dengan nama metode *Design Object Oriented*. Metode ini menjadikan proses analisis dan design ke dalam empat tahapan interatif, yaitu: identifikasi kelas-kelas dan objek-objek, identifikasi semantik dari hubungan objek dan kelas tersebut, perincian *interface* dan implementasi.Keunggulan metode Booch adalah pada detail dan kayanya dengan notasi dan elemen. Pemodelan OMT yang dikembangkan oleh Rumbaugh didasarkan pada analisis terstuktur dan pemodelan *entity-relationship*. Tahapan utama dalam metodologi ini adalah analisis, disain sistem, desain objek dan implementasi. Keunggulan metode ini adalah dalam penotasian yang mendukung semua konsep OO. Metode OOSE dari Jacobson lebih memberi penekanan dan *use case*. OOSE memiliki tiga tahapan yaitu membuat model *requirement* dan analisis, desain dan implementasi dan model pengujian (test Model). Keunggulan

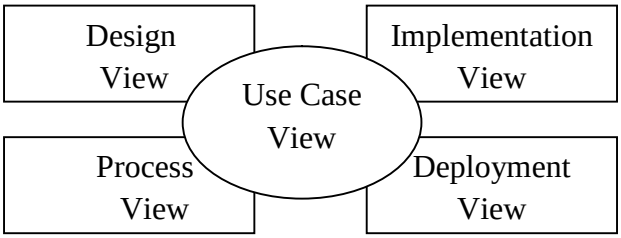
metode ini adalah mudah dipelajari karena memiliki notaasi sederhana namun mencakup seluruh tahapan dalam rekayasa perangkat lunak.

Dengan UML, metode Booch, OMT dan OOSE digabungkan dengan elemen-elemen dari metode lain yang lebih efektif dan elemen-elemen baru yang belum ada pada metode terdahulu sehingga UML lebih ekspresif dan seragam dari pada metode lainnya.

UML adalah hasil kerja dari konsorsium berbagai organisasi yang berhasil dijadikan sebagai standar baku dalam OOAD ( *Object Oriented Analysis* dan *Design* ). UML tidak hanya domain dalam penotasian dilingkungan OO tetapi juga populer di luar lingkungan OO. Ada tiga karakter penting yang melekat di UML yaitu sketsa, cetak biru dan bahasa *pemrograman*. Sebagai sebuah sketsa UML bisa berfungsi sebagai sebuah cetak biru karena sangat lengkap dan detil. Dengan cetak biru ini maka akan bisa diketahui informasi detil tentang coding program (*Forward engineering*) atau bahkan membaca program dan mengimplementasikannya kembali ke dalam diagram (*reverse engineering*). *Reverse engineering* sangat berguna pada situasi dimana kode program yang tidak terdokkumentasi asli hilang atau bahkan belum perna dibuat sama sekali. Sebagai bahasa pemrograman, UML dapat diterjemahkan diagram yang ada di UML menjadi kode program siap untuk dijalankan.

UML dibangun atas model 4+1 *view*. Model ini didasarkan pada fakta bahhwa struktur sebuah sistem dideskripsikan dalam *view* dimana salah satu diantaranya *use case view*. *use case view* ini memegang peran khusus untuk

mengintegrasikan *content* ke *view* yang lain. Model 4+1 *view* ditunjukkan pada gambar II.4




**Gambar II.4 Model 4+1 *View***

(Sumber : Munawar, 2005 : 20)

Kelima *view* tersebut tidak berhubungan dengan diagram yang dideskripsikan di UML. Setiap *view* berhubungan dengan perspektif tertentu dimana sistem akan diuji. *View* yang berbeda akan menekankan pada aspek yang berbeda dari sistem yang mewakili tentang sistem bisa dibentuk dengan menggabungkan informasi-informasi yang ada pada kelima *view* tersebut.

*Use case view* mendefinisikan perilaku eksternal sistem. Hal ini menjadi daya tarik bagi *end user*, analis dan tester. Pandangan ini mendefenisikan kebutuhan sistem karena mengandung semua *view* yang lain yang mendeskripsikan aspek-aspek tertentu dari peran dan sering dikatakan yang mendrive proses perkembangan perangkat lunak.

*Design view* mendeskripsikan struktur logika yang mendukung fungsi-fungsi yang dibutuhkan di *use case*. *Design view* ini berisi definisi komponen program, class-class utama bersama-sama dengan spesifikasi data, perilaku dan interaksinya. Informasi yang terkandung di *view* menjadi pergantian para

progremer karena menjelaskan secara detil bagaimana fungsionalitas sistem akan diimplementasikan.

Implementasi *view* menjelaskan komponen-komponen visi yang akan dibangun. Hal ini berbeda dengan komponen logic yang dideskripsikan pada *design view*. Termasuk disini diantaranya *file exe*, *library* dan *database*. Informasi yang ada di *view* dan integrasi siistem.

Proses *view* berhubungan dengan hal-hal yang berkaitan dengan *concurrency do* dalam sistem. Sedangkan *deployment view* menjelaskan bagaimana komponen-komponen fisik didistribusikan ke lingkungan fisik seperti jaringan komputer dimana sistem akan dijalankan. Kedua view ini menunjukan kebutuhan non fungsional dari sistem seperti toleransi kesalahan dan hal-hal yang berhubungan dengan kinerja (Munawar;2005:17-21).

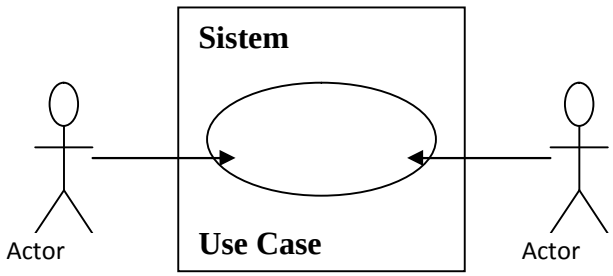
### II.6.1. Use Case Diagram

*Use case* adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara deskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem yang disebut *scenario*. Setiap *scenario* mendeskripsikan urutan kejadian. Setiap urutan diinisialisasi oleh orang, sistem yang lain, perangkat keras dan urutan waktu. Dengan demikian secara singkat bisa dikatakan *use case* adalah serangkaian *scenario* yang digabungkan bersama-sama oleh pengguna tujuan umum pengguna.

Dalam pembicaraan tentang *use case*, pengguna biasanya disebut dengan *actor*. *Actor* adalah sebuah peran yang bisa dimainkan oleh pengguna dalam interaksinya dengan sistem.

Model *use case* adalah bagai dari model *requirement*. Termasuk disini adalah problem domain object dan penjelasan tentang *user interface*. *Use case* memberikan spesifikasi fungsi-fungsi yang ditawarkan oleh sistem dari *persfectif user*.

Notasi *use case* menunjukkan 3 aspek dari sistem yaitu *actor use case* dan *system / sub system boundary. Actor* mewakili peran orang, *system* yang lain atau alat ketika berkomunikasi dengan *use case*. Ilustrasi *actor*, *use case* dan *system* ditunjukkan pada gambar II.5




**Gambar II.5 *Use Case* Diagram**

(Sumber : Munawar, 2005 : 64)

Untuk mengidentifikasi *actor*, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks targer sistem. *Actor* adalah *abstraction* dari orang dan sistem yang lain mengaktifkan fungsi dari target

sistem. Orang atau sistem bila muncul dalam beberapa peran. Perlu dicatat bahwa *actor* berinteraksi dengan use case, tetapi tidak memiliki kontrol atas use *case*.

*Use case* adalah abstraksi dari interaksi antara sistem dan *actor*. Oleh kerena itu sangat penting untuk memilih abstraksi yang cocok. Use case dibuat berdasarkan keperluan actor. Use case harus merupakan ’apa’ yang dikerjakan software aplikasi, bukan ’bagaimana’ software aplikasi mengerjakannya. Setiap use case harus diberi nama yang menyatakan apa hal yang dicapai dari hasil interaksinya dengan actor. Namun use case boleh terdiri dari beberapa kata dan tidak boleh ada dua use case yang memiliki nama yang sama (Munawar ; 2005 : 63-66).

### II.6.2. Class Diagram

Diagram kelas atau class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem.

Kelas memiliki apa yang disebut *Atribut* dan metode atau operasi :

1. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas
2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas

Susunan kelas suatu sistem yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut:

1. Kelas main, kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
2. Kelas yang menangani tampilan sistem, kelas yang mendefinisikan dan mengatur tampilan ke pemakai.

3. Kelas yang diambil dari pendefinisian *use case*, kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*.
4. Kelas yang diambil dari pendefinisian data, kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

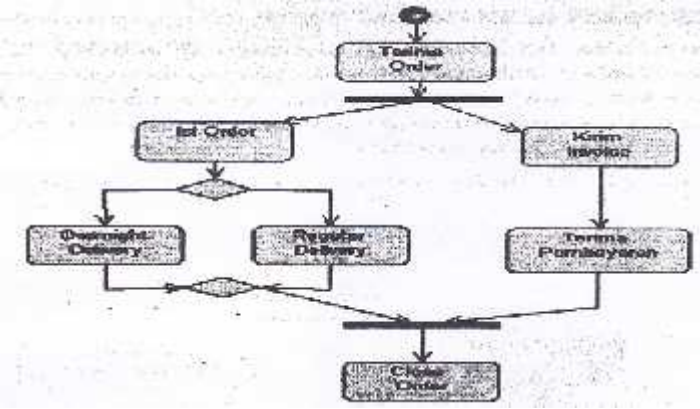
Jenis-jenis kelas diatas juga dapat digabungkan satu sama lain sesuai dengan pertimbangan yang dianggap baik asalkan fungsi-fungsi yang sebaiknya ada pada struktur kelas tetap ada. Susunan kelas juga dapat ditambahkan kelas utilitas seperti koneksi ke basis data, membaca *file* teks, dan lain sebagainya sesuai kebutuhan.

Dalam mendefinisikan metode yang ada di dalam kelas perlu memperhatikan apa yang disebut dengan *cohension* dan *coupling*. *Cohension* adalah ukuran seberapa dekat keterkaitan instruksi di dalam sebuah metode terkait satu sama lain sedangkan *coupling* adalah ukuran seberapa dekat keterkaitan instruksi antara metode yang satu dengan yang lain dalam sebuah kelas. Sebagai aturan secara umum maka sebuah metode yang dibuat harus memiliki kadar *cohesion* yang kuat dan kadar *coupling* yang lemah (Rosa A.S dan M. Shalahuddin ; 2011 : 122-123).

### II.6.3. Activity Diagram

*Activity diagram* adalah teknik untuk mendeskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity diagram* mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaanya dengan *flowchart* adalah

*activity diagram* bisa mendukung perilaku paralel sedangkan flowchart tidak bisa(Munawar ; 2005 : 87). Contoh *activity diagram* sederhana ditunjukkan pada gambar II.6




**Gambar II.6 Contoh *Activity Diagram* Sederhana**

(Sumber :Munawar, 2005 : 111)

### II.6.4. Squence Diagram

*Squence diagram* digunakan untuk menggambarkan perilaku pada sebuah sekenario. Diagram ini menunjukkan sebuah contoh objek dan pesan yang diletakkan diantara objek-objek ini didalam *use case*.

Komponen utama *Squence diagram* terdiri dari atas objek yang dituliskan dengan kotak segiempat bernama. *Messege* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan *progress vertical* (Munawar ; 2005 : 109).

1. Objek / *participant*

Objek diletakkan di dekat bagian atas diagram dengan urutan dari kiri ke kanan. Mereka diatur dalam urutan guna menyederhanakan diagram. Setiap

*participant* dihubungkan garis titik-titik yang disebut *lifeline.* Sepanjang *lifeline* ada kotak yang disebut *activation*. *Activation* mewakili sebuah eksekusi operasi dari *participant*. Panjang kotak ini berbanding lurus dengan durasi *activation*. *Activation* mewakili sebuah eksekusi operasi dari participant. Panjang kotak ini berbanding lurus dengan durasi *activation.* Bentuk *participant* dapat dilihat pada gambar II.7

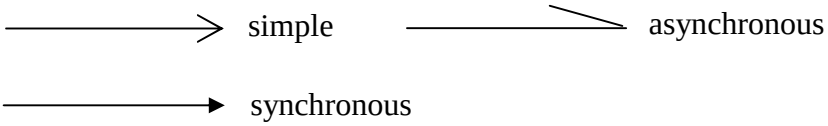
**Gambar II.7 Bentuk *Participant***

(Sumber :Munawar, 2005 : 88)

2. *Messege*

Sebuah *messege* bergerak dari suatu *participant* ke *participant* yang lain dan dari *lifeline* ke *lifeline* yang lain. Sebuah *participant* bisa mengirim sebuah *message* kepada dirinya sendiri.

Sebuah *message* bisa jadi *simple*, *synchronous* atau *asynchoronous*. *Message* yang *simple* adalah sebuah perpindahan (transfer), contoh dari satu *participant* ke *participant* yang lainnya. Jika suatu *participant* mengirimkan sebuah *message* tersebut akan ditunggu sebelum di proses dengan urusannya. Namun jika *message asynchoronous* yang dikirimkan, maka jawabannya atas *message* tersebut tidak perlu ditunggu. Simbol *message* pada *squnence diagram* dapat dilihat pada gambar II.8



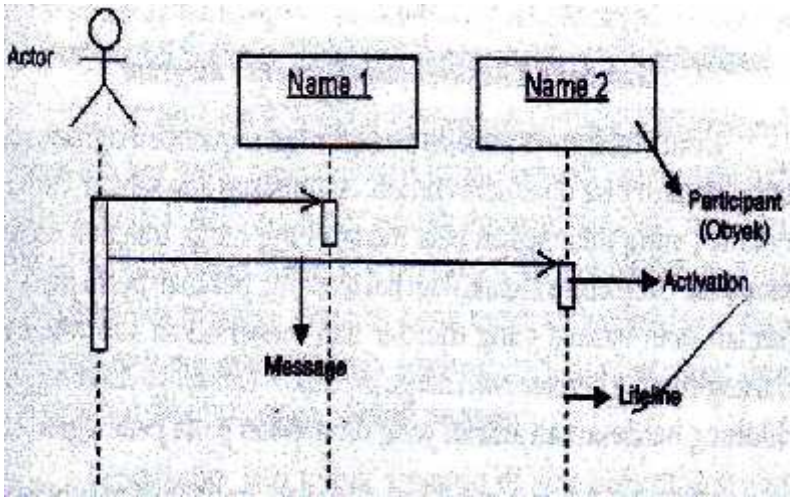

**Gambar II.8 Bentuk *Messege***

(Sumber :Munawar, 2005 : 88)

3. *Time*

*Time* adalah diagram yang mewakili waktu pada arah vertikal. Waktu dimulai dari atas ke bawah. *Message* yang lebih dekat dari atas akan dijalankan terlebih dahulu dibanding *message* yang lebih dekat kebawah.

Terdapat dua dimensi pada *squence diagram* yaitu dimensi dari kiri ke kanan menunjukkan tata letak participant dan dimensi dari atas ke bawah menunjukkan lintasan waktu. Simbol-simbol yang ada pada *squence diagram* ditunjukkan pada gambar II.9




**Gambar II.9 entuk *Time***

(Sumber :Munawar, 2005 : 89)