# BAB II

# TINJAUAN PUSTAKA

## II.1 Keamanan Komputer

Sistem keamanan komputer digunakan untuk menjamin agar sumber daya tidak digunakan untuk mnamin agar sumber daya tidak digunakan atau dimodifikasi orang yang tidak diotorisasi. Pengamanan termasuk masalah teknis, manajerial, legalitas dan politis.

Keamanan sistem terbagi menjadi tiga, yaitu :

1. Keamanan eksternal adalah pengamanan yang berhubungan dengan fasilitas komputer dari penyusup dan bencana, misalya bencana alam.
2. Keamanan *interface* pemakai, berkaitan dengan identifkasi pemakai sebelum dijinkan mengakses program dan data yang tersimpan didalam sistem.
3. Keamanan internal, berkaitan dengan beragam pengamanan yang dibangun pada perangat keras dan sistem operasi untukmenjamin operasi yang handal dan untuk menjaga keutuhan program serta data

Sehubungan dengan keamanan ini terdapat dua masalah yang penting, yaitu :

1. Kehilangan data,yang dapat disebabkan antara lain oleh:
    a. Bencana, seperti kebakaran, banjir, gempa bumi, perang, maupun kerusuhan.
    b. Kesalahan perangkat keras dan perangkat lunak yang disebabkan oleh tidak berfungsinya pemroses, *disk* yang tidak terbaca, kesalahan telekomunikasi, dan kesalahan program (*bugs*).

c. Kesalahan manusia, seperti salah dalam memasukkan data, salah memasang *disk*, eksekusi program yang salah, dan kehilangan *disk*.

2. Penyusup (*intruder*) terdiri dari :

a. Penyusup pasif, yaitu membaca data yang tidak diotorisasi.

b. Penyusup aktif, yaitu mengubah data yang tidak diotorisasi.

Ketika hendak merancang sebuah sistem yang aman dari serangan para *intruder* adalah penting untuk mengetahui sistem tersebut akan dilindungi dari *intruder* apa. Dibawah ini ada empat contoh tujuan *intruder* dalam melakukan serangan :

1. Keingintahuan seseorang akan hal-hal pribadi orang lain.

Banyak orang yang mepunyai *PC* yang terhubung kejaringan.berapa orang dalam jaringan tersebut suka mebaca *e-mail* dan *file* orang lain jika didalam jaringan tersebut tidak ditempatkan sistem penghalang. Sebagai contoh, sebagian besar sistem *UNIX* mempunyai default bahwa semua *file* yang baru diciptakan dapat dibaca orang lan.

2. Penyusup oleh orang–orang dalam.

Pelajar, *programmer*, operator, dan personil teknis menganggap bahwa mematahkan sistem keeamanan komputer lokal merupakan suatu tantangan. Mereka biasanya sangat ahli dan bersedia mengorbankan banyak waktu untuk melakukan hal tersebut.

3. Keinginan untuk mendapatkan uang.

Beberapa pemrogram *bank* mencoba mencuri uang dari *bank* tempat mereka bekerja dengan mengubah *software* sehingga akan memotong bunga dari pada membulatkanya, menyimpan uang kecil untuk mereka sendiri, menarik uang dari *account* yag sudah tidak digunakan selama bertahun-tahun ataupun memeras ("Bayar saya, atau saya akan menghancurkan semua *record* bank Anda.").

4. *E-spionase* komersial atau militer

*E-spionase* adalah usaha serius yang iberi dana besar oleh pesaing atau pihak mush untuk mencuri pogram, rahasia dagang, ide-ide paten, teknologi, rencana bisnis, dan sebagainya. Seringkai usaha ini melibatkan *wiretapping*, dimana antenna diarahkan ke suatu program untuk menangkap radiasi elektromagnetis yang memancar dari komputer itu.

### II.1.1 Ancaman Keamanan

Target pengamanan adalah menghidari, mencegah dan mengatasi ancaman-ancaman terhadap sistem. Kebutuhan akan pengamanan komputer dapat dikategorikan dalam tiga aspek, yaitu :

1. Kerahasian, dimana informasi pada sistem komputer itu terjamin kerahasiaannya, hanya dapat diakses oleh pihak-pihak yang diotorisasi, keutuhan serta konsistensi data pada sistem tersebut tetap terjaga.
2. Integritas, di mana sumber daya sistem terjamin hanya dapat dimodifikasi oleh pihak-pihak yang diotorisasi.

3. Ketersediaan, adalah sumber daya sistem komputer terjamin akan tersedia bagi pihak-pihak yang diotorisasi pada saat diperlukan.

### II.1.2 Tipe-tipe Ancaman Keamanan

Fungsi-fungsi ancaman komputer dapat digunakan sebagai dasar untuk menentukan model tipe ancaman dari suatu sistem computer. Berdasarkan fungsi sistem komputer sebagai penyedia informasi, ancaman terhadap terhadap sistem komputer dikategorikan menjadi empat, yaitu :

1. *Interruption*, merupakan satu ancaman terhadap *avaibability*, informasi atau data yang ada dalam sistem komputer dirusak, dihapus, sehingga jika dibutuhan maka sudah tidak ada lagi.
2. *Interception*, merupakan ancaman terhadap kerahasian (*secrecy*).
   Informasi yang ada didalam sistem disadap oleh orang yang tidak berhak.
3. *Modification*, merupakan ancaman terhadap integritas. Orang yang tidak berhak berhasil menyadap lalu lintas informasi yang sedang dikirim lalu mengubahnya sesuai keinginan orang itu.
4. *Fabrication*, merupakan ancaman terhadap integritas. Orang yang tidak berhak berhasil meniru atau memalsukan suatu informasi sehingga orang yang menerima informasi tersebut menyangka informasi tersebut berasal dari orang yang dikehendaki oleh si penerima informasi tersebut. (Sumber : Pengamanan Sistem Komputer, Janner Simarmata,2006 :5-11)

## II.2 Pengertian Bahan Ajar

Bahan ajar adalah segala bentuk konten baik teks, *audio*, foto, *video*, animasi, dan lain-lain yang dapat digunakan untuk belajar. Ditinjau dari subjeknya, bahan ajar dapat dikatogorikan menjadi dua jenis, yakni bahan ajar yang sengaja dirancang untuk belajar dan bahan yang tidak dirancang namun dapat dimanfaatkan untuk belajar. Banyak bahan yang tidak dirancang untuk belajar, namun dapat digunakan untuk belajar, misalnya kliping koran, *film*, sinetron, iklan, berita, dan lain-lain. Karena sifatnya yang tidak dirancang, maka pemanfaatan bahan ajar seperti ini perlu diseleksi sesuai dengan tujuan pembelajaran.

Bahan belajar yang dirancang adalah bahan yang dengan sengaja disiapkan untuk keperluan belajar. Ditinjau dari sisi fungsinya, bahan ajar yang dirancang dapat dikelompokkan menjadi tiga kelompok, yaitu bahan presentasi, bahan referensi, dan bahan belajar mandiri. Sedangkan ditinjau dari media, bahan ajar dapat kelompokkan menjadi bahan ajar cetak, *audio*, *video*, televisi, *multimedia*, dan *web*. Sekurang-kurangnya ada empat ciri bahan ajar yang sengaja dirancang, yakni:

1. adanya tujuan yang jelas,
2. ada sajian materi,
3. ada petunjuk belajar, dan
4. ada evaluasi keberhasilan belajar

### II.2.1 Unsur-unsur Bahan Ajar

Bahan ajar setidak tidaknya harus memiliki enam unsur, yaitu mencakup tujuan, sasaran, uraian materi, sistematika sajian, petunjuk belajar, dan evaluasi. Sebuah bahan ajar harus mempunyai tujuan. Tujuan harus dirumuskan secara jelas dan terukur mencakup kriteria ABCD *(audience, behavior, criterion, dan degree).* Sasaran perlu dirumuskan secara spesifik, untuk siapa bahan relajar itu ditujukan. Sasaran bukan sekedar mengandung pernyataan subjek orang, Namur juga harus mencakup kemampuan apa yang menjadi prasyarat yang harus sudah mereka kuasai agar dapat memahami bahan ajar ini. (Sumber : Pengembangan Bahan Ajar Berbasis Web, Lu'mu Tasri, 2011)

## II.3 Pengenalan *Mime Base64*

*Mime* Base64 atau quadrosexagesimal adalah penempatan notasi dengan menggunakan bilangan *radix* 64. *Base64* merupakan bilangan berbasis 2 (*poweroftwo*) terbesar yang dapat direpresentasikan dengan menggunakan karakter *ASCII*. Hal ini memungkinkan untuk penggunaannya melakukan *encoding* terhadap *email* dan lainnya. Base64 menggunakan karakter A – Z, a – z dan 0 – 9 untuk 62 nilai pertama, sedangkan 2 nilai terakhir digunakan untuk symbol (+ dan /). Beberapa metode *encoding* lain seperti *uuencode* dan *binhex* menggunakan 64 karakter yang berbeda untuk mewakili *6 binary digit*, namun metode-metode tersebut tidak disebut sebagai *encoding Base64.*

**Tabel II.1 Encoding Base64**

| Base64 Encoding Table | | | | | | | |
|---|---|---|---|---|---|---|---|
| Value | Char | Value | Char | Value | Char | Value | Char |
| 0 | A | 16 | Q | 32 | g | 48 | W |
| 1 | B | 17 | R | 33 | h | 49 | X |
| 2 | C | 18 | S | 34 | i | 50 | Y |
| 3 | D | 19 | T | 35 | j | 51 | Z |
| 4 | E | 20 | U | 36 | k | 52 | 0 |
| 5 | F | 21 | V | 37 | l | 53 | 1 |
| 6 | G | 22 | W | 38 | m | 54 | 2 |
| 7 | H | 23 | X | 39 | n | 55 | 3 |
| 8 | I | 24 | Y | 40 | o | 56 | 4 |
| 9 | J | 25 | Z | 41 | p | 57 | 5 |
| 10 | K | 26 | a | 42 | q | 58 | 6 |
| 11 | L | 27 | b | 43 | r | 59 | 7 |
| 12 | M | 28 | c | 44 | s | 60 | 8 |
| 13 | N | 29 | d | 45 | t | 61 | 9 |
| 14 | O | 30 | e | 46 | u | 62 | + |
| 15 | P | 31 | f | 47 | v | 63 | / |

(Salman Teguh, 5, 2007)

### II.3.1 Penggunaan *Mime Base64*

Berikut ini adalah contoh penggunaan dari *MIME Base 64* dalam melakukan encoding karakter.

Kutipan dari Thomas Hobbes's Leviathan :

"*Man is distinguished, not only by his reason, but by this singular passion from other animals, which is a lust of the mind, that by a perseverance of delight in the continued and indefatigable generation of knowledge, exceeds the short vehemence of any carnal pleasure."*

Hasil dari encoding dengan menggunakan MIME Base64 :

*"TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieSB0aGlzIHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGljaCBpcyBhIGx1c3Qgb2YgdGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbiB0aGUgY29udGludWVkIGFuZCBpbmRlZmF0aWdhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdlLCBleGNlZWRzIHRoZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS4="*

Pada hasil encoding diatas kata "*Man*" diganti menjadi "TWFu". Pada table ASCII huruf M, a, n disimpan sebagai 77, 97, 110 atau dengan kata lain 010001101, 01100001, 01101110 pada bilangan berbasis 2. Apabila ketiga byte tersebut digabungkan, maka akan dihasilkan 24 *bit buffer* yaitu 010001101011000010110110. Angka tersebut harus dikonversi sehingga berbasis 64, caranya dengan membagi 24 *bit* tersebut dengan 6. Maka dihasilkan 4 bagian dengan masing-masing 6 *bit*. Kemudian masingmasing bagian tersebut dikonversi ke nilai yang ada di *Base64*.

**Tabel II.2 Hasil konversi dari *ASCII* ke *Base64* untuk kata "Man"**

| Huruf | M | | | | | | | | A | | | | | | | | n | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ASCII | 77 | | | | | | | | 97 | | | | | | | | 110 | | | | | | | |
| Bit | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| Index | 19 | | | | | | | | 22 | | | | | | | | 5 | | | | | | | |
| Base 64 Encoded | T | | | | | | | | W | | | | | | | | F | | | | | | | |
| | | | | | | | | | | | | | | | | | u | | | | | | | |

(Salman Teguh, 2007, 6)

Proses *padding* akan dilakukan apabila sekelompok karakter yang dimiliki tidak bernilai 3 Byte (24 bit). *Padding* dilakukan dengan menambahkan karakter '='. Contoh penggunaan *padding* dapat dilihat pada tabel berikut.

**Tabel II.3 Proses *padding* 2 *Byte nilai 0***

| Huruf | ! | | | |
|---|---|---|---|---|
| ASCII | 33 | | | |
| Bit | 0 0 1 0 0 0 0 1 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | |
| Index | 8 | 16 | | |
| Base 64 Encoded | I | Q | = | = |

(Salman Teguh, 2007, 6)

Apabila terdapat *singlebyte* maka jumlah *padding* yang ditambahkan adalah 2 Byte yang bernilai 0. Sehingga memenuhi aturan 3 *Byte* (24 *bit*), seperti dapat dilihat pada tabel II.3. Sedangkan pada tabel II.4 jumlah *byte padding* yang ditambahkan adalah 1 *Byte* karena sebelumnya telah memiliki 2 *Byte*.

**Tabel II.4 Proses *padding* 1 *Byte* nilai 0**

| Huruf | ! | S | | |
|---|---|---|---|---|
| ASCII | 105 | 115 | | |
| Bit | 0 1 1 0 1 0 0 1 | 0 1 1 1 0 0 1 1 | 0 0 0 0 0 0 0 0 | |
| Index | 26 | 23 | 12 | |
| Base 64 Encoded | A | X | M | = |

(Salman Teguh, 2007, 6)

## II.4 Pengenalan PHP

PHP merupakan singkatan dari *PHP Hypertext Preprocessor* yaitu bahasa berbentuk skrip yang disimpan dan dijalankan pada sisi *server*. Sedangkan hasil (*output*) eksekusi dari *server* ditampilkan pada *client* dengan menggunakan *web browser*. Penggunaan PHP biasanya difokuskan pada pengembangan aplikasi yang server side scripting. Namun sebenarnya terdapat beberapa area utama penggunaan PHP, diantaranya :

1. *Server Side Scripting*
2. *Command Line Scripting*
3. *Desktop Application* (Menggunakan PHP-GTK)

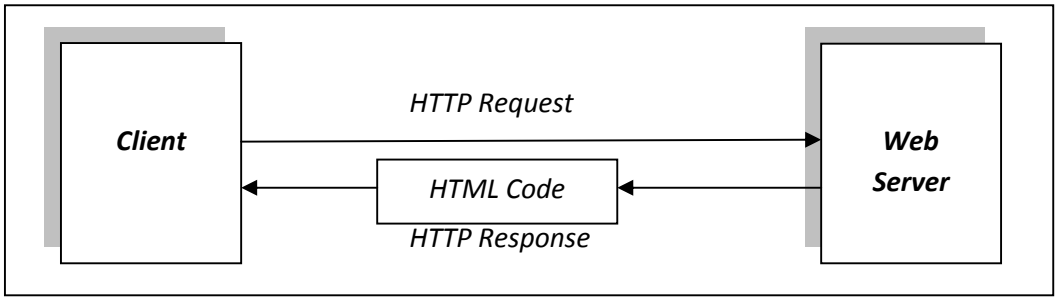
Dari semua penggunaan PHP, penggunaan pada *serverside scripting* merupakan yang paling sering digunakan. Terutama bila membutuhkan website atau aplikasi berbasis *web* yang dinamis.

PHP dapat dijalankan pada hampir semua sistem operasi yang ada saat ini, diantaranya *: Linux, Unix* (*HPUX, Solaris, OpenBSD*), *Microsoft Windows*, Mac *OS X, RISC OS*, dan lain-lain. PHP juga mendukung banyak *web server*, diantaranya : *Apache, Microsoft Internet Information Server* (IIS), *Personal Web Server* (PWS), *Netscape Server, iPlanet server, Oreilly Website Pro Server, Caudium, Xitami, OmniHTTPD*, dan lainlain.

Sedangkan dalam melakukan penulisan program PHP dapat menggunakan *procedural programming* atau *object oriented programming*. Selain itu dapat juga menggunakan gabungan keduanya. Dengan menggunakan PHP *output* (hasil

keluaran) tidak harus berupa HTML, namun PHP mempunyai kemampuan untuk menghasilkan gambar, PDF bahkan file Flash yang dihasilkan secara *on fly*.

### II.4.1 Konsep Kerja PHP




**Gambar II.1 Konsep Kerja *HTML***

(Salman Teguh, 2007, 3)

Secara umum konsep kerja dari PHP hampir sama dengan konsep kerja dari HTML. Dimana terdapat *client* yang meminta (*request*) berkas tertentu yang disimpan pada sisi server. Kemudian *server* tersebut mengirimkan berkas tersebut kepada *client*. Perbedaannya adalah pada konsep kerja HTML berkas yang dikirimkan sama dengan yang disimpan pada sisi *server*. Sedangkan pada konsep kerja PHP, berkas yang dikirim merupakan hasil proses pada *server* sehingga bisa terdapat perbedaan antara *source code* yang disimpan pada sisi *server* dengan yang dikirim pada *client*.

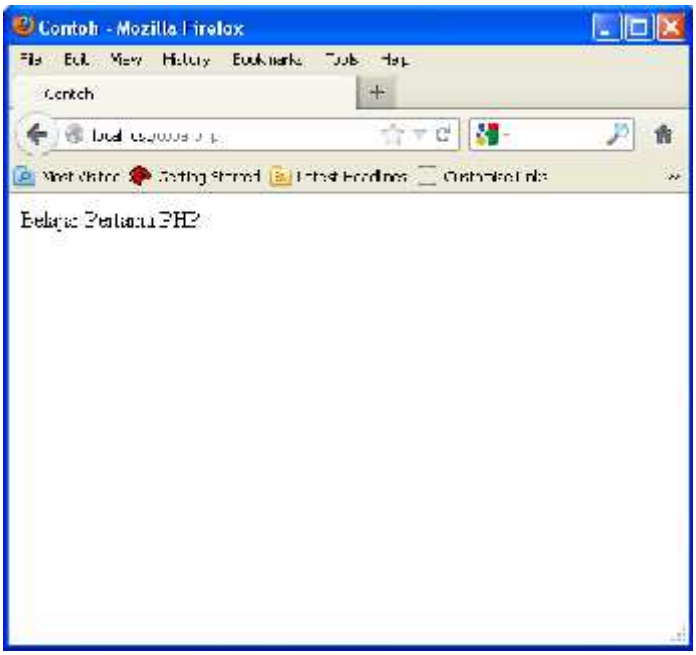
### II.4.2 Skrip PHP

Dalam setiap penulisan kode PHP biasanya diawali dengan penulisan *<?* atau *<?php* dan diakhiri dengan *?>*. Pasangan tersebut dikenal sebagai *tag* kode

PHP. Berdasarkan pada tag tersebut maka sebuah *server* dapat mengenali bahwa skrip yang akan dieksekusi adalah skrip PHP yang kemudian server memprosesnya. Berikut ini adalah contoh skrip PHP (yang disisipkan kedalam HTML) :

```
<html>
    <head><title>Contoh</title></head>
    <body>
        <?php
        Echo  “Belajar Pertama PHP”;
        ?>
    </body>
</html>
```

Pada gambar II.2, merupakan hasil keluaran dari skrip PHP diatas yang telah dijalankan (eksekusi) oleh web server.




**Gambar II.2 Contoh Keluaran Eksekusi *PHP***

(Salman Teguh, 2007, 4)

### II.4.3 Menggunakan kontrol dalam PHP

**1. Pernyataan If**

Sebagaimana telah Anda pelajari pada JavaScript, pernyataan if pada PHP juga berguna untuk melakukan pengambilan keputusan terhadap lebih dari satu alternatif. Bentuk pernyataan if yang tersedia pada PHP adalah :

1. If
2. If..else
3. If..elseif

**2. Bentuk Pernyataan If Sederhana**

Bentuk if yang paling sederhana adalah seperi berikut :

If (ekspresi)

Pernyataan

Pada bentuk ini, bagian pernyataan akan dijalankan kalau bagian ekspresi bernilai benar. Sekiranya jumlah pernyataan yang akan dijalankan oleh if lebih dari satu, maka Anda bisa menuliskannya seperti berikut :

```
If(ekspresi){
Pernyataan_1;
...
Pernyataan_n;
}
```

(Abdul Kadir, 2009: 269-272)

### 3. Bentuk if-else

Bentuk kedua pernyataan *if* melibatkan bagian *else*. Formatnya seperti berikut:

```
If(ekspresi)
    Pernyataan_1;
Else
    Pernyataan_2;
```

Pada bentuk ini,

1. Bagian pernyataan_1 dijalankan kalau ekspresi bernilai benar, dan
2. Bagian pernyataan_2 dijalankan kalau ekspresi bernilai salah.

Bentuk if-else juga bisa berupa :

```
If(ekspresi)
{
Pernyataan_1;
}
Else
{
Penyataan_2;
}
```

Keadan di depan terjadi sekiranya baik bagian pernyataan_1 maupun pernyataan_2 mengandung sejumlah pernyataan. Tanda {} bisa tidak disertakan kalau didalamnya hanya ada satu penyataan.

## II.5 Pengertian UML

UML singkatan dari Unified Modeling Language yang berarti bahasa pemodelan standar. (Cloneles, 2003 : bab1) mengatakan sebagai bahasa, berarti UML memiliki sintaks dan semantik. Ketika kita membuat model mengguunakan konsep UML ada aturan - aturan yang harus diikuti. Bagaimana elemen pada model – model yang kita buat berhubungan satu dengan lainnya harus mengikuti standar yang ada. UML bukan hanya sekedar diagram, tetapi juga menceritakan konteksnya. Ketika pelanggan memesan sesuatu dari system, bagaimana transaksinya? Bagaimana system mengatasi error yang terjadi? Bagaimana keamanan terhadap system yang kita buat? Dan sebagainya dapat dijawab dengan UML. (Sumber : Prabowo Pudjo Widodo dan Heriawanti, 2006: 7)

UML diaplikasikan untuk maksud tertentu, biasanya antara lain untuk:

1. Merancang Perangkat Lunak
2. Sarana komunikasi antara perangkat lunak dengan proses bisnis
3. Menjabarkan system secara rinci untuk analisa dan mencari apa yang diperlukan system.
4. Mendokumentasi system yang ada, proses – proses dan organisasinya.

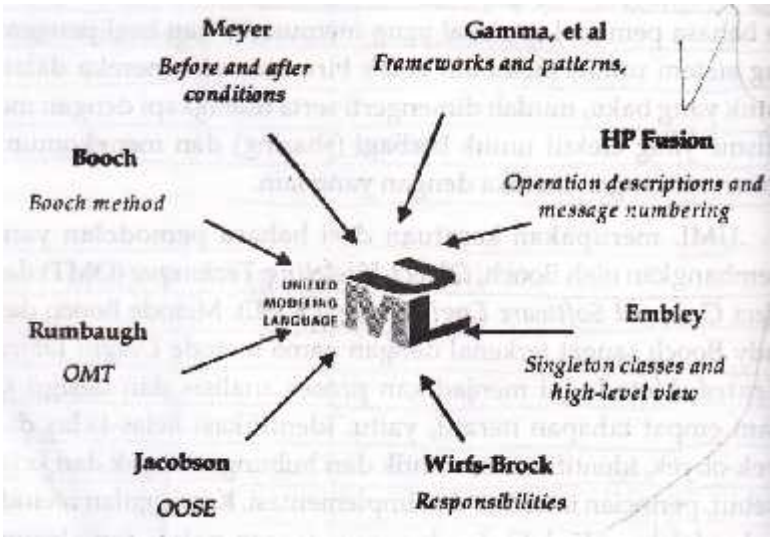
UML telah diaplikasikan dalam bidang investasi perbankan, lembaga kesehatan, departemen pertahanan, system terdistribusi, system pendukung alat kerja, retail, sales dan supplier.

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 iteratif, yaitu: identifikasi kelas-kelas dan objek-objek, identifikasi semantik dari hubungan objek dan kelas tersebut, perincian interface dan implementasi. Keunggulan metode Booch adalah pada detil dan kayanya dengan notasi dan elemen. Pemodelan OMT yang dikembangkan oleh Rumbaugh didasarkan pada analisis terstruktur dan pemodelan *entity-relationship*. Tahapan utama dalam metodologi in adalah analisis, desain sistem, desain objek dan impelmentasi. Keunggulan motode 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 notasi yang sederhananamun mencangkup seluruh tahapan dalam rekayasa perangkat lunak.

Dengan UML, metode Booch, OMT dan OOSE digabungkan dengan membuang elemen-elemen yang tidak praktis ditambah dengan elemen-elemen dari metode lain yang lebih efektif dan elemen-elemen baruyang belum ada pada

metode terdahulu sehingga UML lebih ekspresif dan seragam daripada metode lainnya. Unsur-unsur yang membentuk UML ditunjukkan dalam Gambar II.3



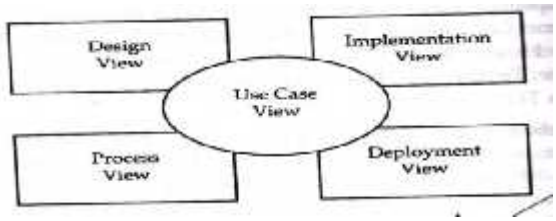

**Gambar II.3 Unsur-unsur yang membentuk UML**

( Munawar, 2005:18)

UML adalah hasil kerja dari konsorsium berbagai organisasi yang berhasil dijadikan sebagai standar baku dalam OOAD (*Object Oriented Analysis dan Design* ). UML tidak hanya dominan dalam penotasian di lingkungan 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 kerena sangat lengkap dan detil. Dengan cetak biru ini maka akan bisa diketahui informasi detil tentang coding program (*Forward engineering* ) atau bahkan membaca program dan mengiterpretasikannya kembali ke dalam diagram (*reverse engineering* ). *Reverse engineering* sangat berguna pada situasi dimana kode program yang tidak

terdokumentasi asli hilang atau bahkan belum dibuat sama sekali. Sebagai bahasa pemrogaraman, UML dapat menterjemahkan diagaram yang ada di UML menjadi kode program siap untuk dijalankan.

UML dibangun atas model 4+1 *view*. Model ini didasarkan pada fakta bahwa 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***

(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 tari bagi *end user*, analis dan tester. Pandangan ini mendefinisikan kebutuhan sistem karena mengandung semua *view* yang lain yang mendeskripsikan aspek-aspek

tertentu dari peran dan sering dikatakan yang mendrive proses pengembangan 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, perilkau dan interaksinya. Informasi yang terkandung di *view* ini menjadi perjatian para programer karena menjelaskan secara detil bagaimana fungsionalitas sistem akan diimplementasikan.

Implemantasi *view* menjelaskan komponen-komponen fisi dari sistem 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 sistem.

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

### II.5.1 Use Case Diagram

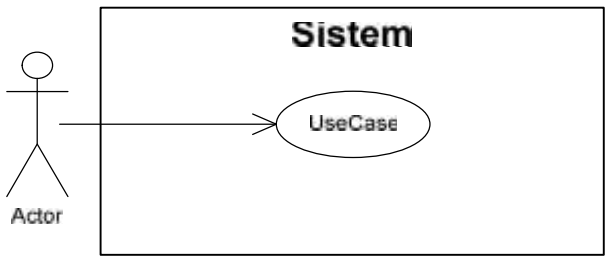
*Use case* adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara deskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita

bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem disebut *scenario.* Setiap *scenario* mendeskripsikan urutan kejadian. Setiap urutan diinisialisasi oleh orang, sistem yang lain, perangkat keras atau urutan waktu. Dengan demikian secara singkat bisa dikatakan *use case* adalah serangkaian *scenario* yang digabungkan bersama-sama oleh 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 *perspectif 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*, *usecase* dan *system* ditunjukkan pada gambar II.5




**Gambar II.5 *Usecase* Diagram**

(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 yang mengaktifkan fungsi dari target sistem. Orang atau sistem bisa 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 karena 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 interaaksinya 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.5.2 Class Diagram

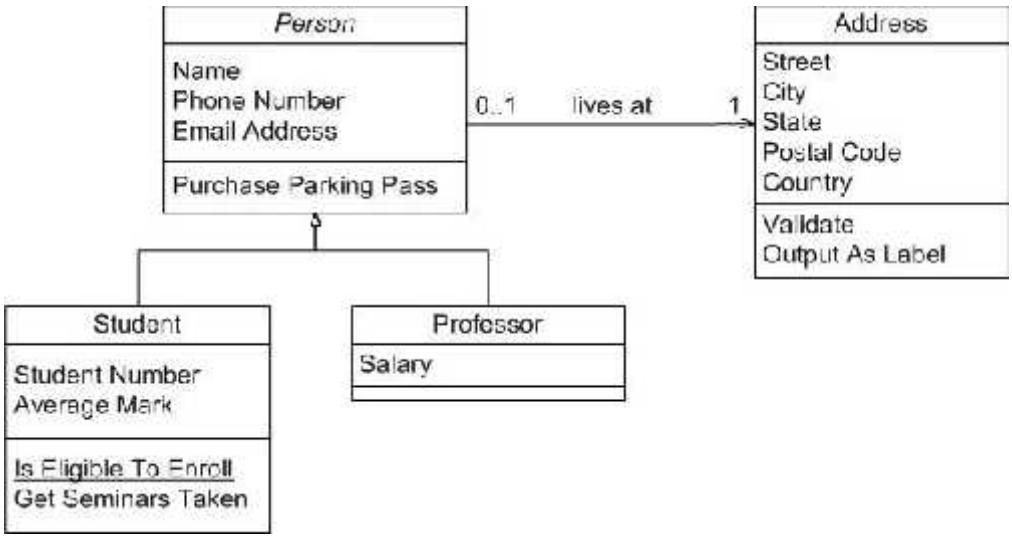
*Class* adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (*atribut*/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).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

1. Namakelas
2. Atribut
3. Metode

*Atribut* dan metode dapat memiliki salah satu sifa tberikut :

1. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
2. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan.
3. *Public*, dapat dipanggil oleh siapa saja.

*Class* dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metode. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class.* Contoh diagram *class* dapat dilihat pada gambar II.6 dibawah ini:

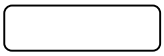
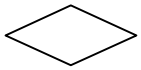
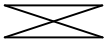
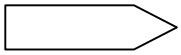


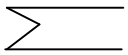

**Gambar II.6 *Class* Diagram**

( Munawar, 2005:220)

### II.5.3 *Activity* Diagram

*Activity* Diagram adalah teknik untuk mendiskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity* Diagram mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaannya dengan *flowchart* adalah *activity* diagram bisa mendukung perilaku paralel sedangkan flowchart tidak bisa.Adapun simbol *activity diagram* dapat dilihat pada table II.5 :

| Notasi | Keterangan |
|---|---|
|  | Titik Awal |
|  | Titik Akhir |
|  | *Activity* |
|  | Pilihan untuk pengambilan keputusan |
|  | *Fork*digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu |
|  | *Rake*menunjukkanadanyadekomposisi |
|  | Tandawaktu |
|  | Tandapengiriman |

| | |
|---|---|
|  | Tandapenerimaan |
|  | AliranAkhir (*Flow Final*) |

**Tabel II.5. Simbol *Activity* Diagram**

(Munawar, 2005:110)

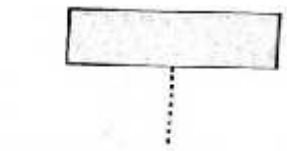
### II.5.4 Squence Diagram

*Sequence diagram* digunakan untuk menggambarkan perilaku pada sebuah sekenario. Diagram ini menunjukkan sejumlah contoh objek dan pesan yang diletakkan diantaran objek-objek ini di dalam *use case.*

Komponen utama *sequence diagram* terdiri atas objek yang ditulisakan dengan kotak segiempat bernama. *Messege* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan *progress vertical.*

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 dengan 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.* Bentuk *participant* dapat dilihat pada gambar II.7



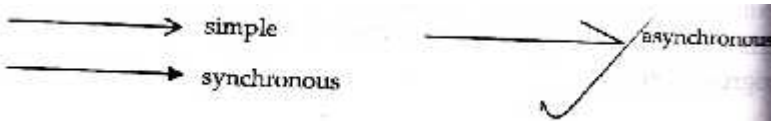
**Gambar II.7 Bentuk *Participant***

(Munawar, 2005:88)

2. *Messege*

Sebuah *messsage* bergerak dari satu *participant* ke *participant* yang lain dan dari satu *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 sebuah *participant*mengirimkan sebuah *messagae* tersebut akan ditunggu sebelum diproses 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



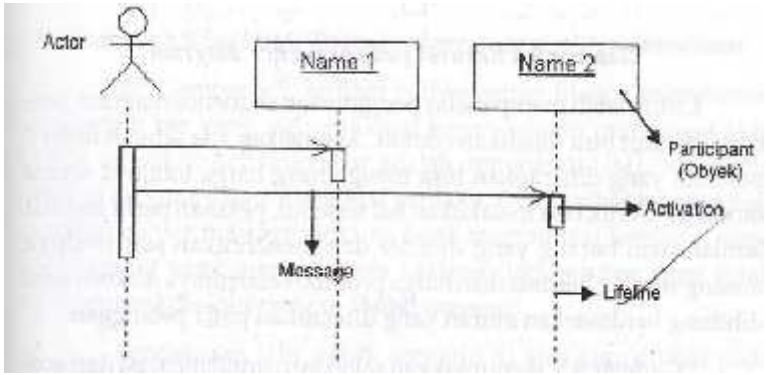

**Gambar II.8 Bentuk *Messege***

(Munawar,2005:88)

*3. Time*

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

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 Bentuk *Messege***

(Munawar,2005:89)