BAB II
LANDASAN TEORI

2.1 Sejarah Telepon Seluler

Konsep dasar teknologi yang dipakai dalam sebuah ponsel sebenarnya merupakan pengembangan dari teknologi radio yang dipadukan dengan teknologi komunikasi telepon. Telepon diciptakan oleh Allexander Graham Bell pada tahun 1976, dan komunikasi tanpa kabel (*wireless*) ditemukan oleh Nikolai Tesla pada tahun 1880 yang kemudian dikembangkan oleh Guglielmo Marconi. Sebelum ponsel diciptakan biasanya radio telepon dipasang ke dalam mobil untuk dapat digunakan sebagai sarana komunikasi secara *mobile*.

Pada sistem radio telepon diperlukan menara antena yang terpusat dan hanya menyediakan 25 saluran pada setiap menaranya. Menara antena ini harus mempunyai daya pancar yang sangat kuat untuk dapat memancarkan sinyal hingga 70 Km, karena saluran yang cukup terbatas mengakibatkan tidak semua orang dapat menggunakan radio telepon ini. Sejak tahun 1940 teknologi *wireless* dan *cellular* mulai diperkenalkan secara komersil, sistem radio seluler ini ternyata telah menjadi proyek nasional di negara Jepang sejak 1967, sejak tahun tersebut Jepang melakukan beberapa eksperimen tentang teknologi perambatan pada radio, kekuatan sinyal dan penelitian tentang daya tangkap sinyal diarea perkotaan menggunakan frekwensi 400 Mhz dan kemudian 900 Mhz.

Uji coba sistem seluler ini termasuk sukses dilaksanakan pada tahun 1965 dan diterapkan dikota metropolitan Tokyo. Salah satu perusahaan yang sangat ambisius dalam hal pengembangan mikroprosesor dan semi konduktor adalah Motorola. Kerja keras dan proyek ambisius ini ternyata membuahkan hasil yaitu sirkuit-sirkuit buatan Motorola digunakan oleh Badan Antariksa Amerika Serikat, pada awal program ruang angkasa dimulai, pada tahun 1973 Motorola mempetakan sistem radio sehingga pada tahun 1973 *Project Manager* Motorola Dr. Couper untuk pertama kalinya memasang *base station* di *network* dan membuat *prototipe* telepon seluler.

Pada tahun 1974 *federal communication commision* (*FCC*) Amerika memutuskan untuk membuka dan menambah spektrum 115 Mhz serta 2300 *chanel* untuk digunakan bagi perkembangan telepon seluler secara komersil. Dengan persetujuan ini maka bisnis radio dan *manufacture* telepon semakin berkembang dengan pesat akhirnya pada tahun 1975 *FCC* memberikan ijin kepada *bell system* untuk mengembangkan dan melakukan uji coba sistem yang telah ditelitinya. *Bell lab* adalah pesaing Motorola dalam hal pengembangan telepon seluler. Sejak tahun 1960 hingga 1970 kedua perusahaan ini bersaing untuk menciptakan penerangan teknologi seluler bagi kehidupan sehari-hari, *Bell system* telah diuji cobakan di Chicago dan Illionis, ternyata *Bell system* ini telah berkembang dengan pesat dan menjadi pelayanan telepon terbaik didunia. *Bell system* melayani kota-kota kecil dan besar di Amerika Serikat dan telah mencapai 80 % dari penduduk Amerika.

2.1.1 *GSM* (*Global System For Mobile Communiction*)

GSM merupakan teknologi selular digital yang dikembangkan pertama kali oleh *group special mobile* di Eropa pada tahun 1982. Teknologi *GSM* memiliki karakteristik yang dapat menangani suara secara efisien namun memiliki keterbatasan dalam kemampuan *transfer* data aplikasi internet. Komunikasi data pada *GSM* dijalin melalui mekanisme *circuit swicthed connection* yang berati hubungan diawali dengan *dial* dari pengguna dan diakhiri dengan pemutusan hubungan, apabila pengguna ingin mengakses data lagi maka mereka harus melakukan *dial* lagi. Untuk memecahkan masalah tersebut maka dibutuhkan teknologi paket data untuk *GSM* menggunakan *packet swicthed connection*.

Pada tahun 2000 muncul teknologi *High Speed Circuit Swicthed Data* (*HSCSD*), teknologi ini memiliki mekanisme *transfer data circuit swicthed* yang mirip dengan *GSM* namun memiliki kelebihan dalam kemampuan untuk menggunakan lebih dari satu *time slot* dari 8 *time slot* untuk satu kali koneksi (*GSM* hanya dapat menggunakan 1 *time slot* untuk satu kali koneksi). Kemampuan ini menjadikan *HSCSD* dapat mencapai kecepatan *transfer* data hingga 57,6 kbps.

Paket data untuk *GSM* diperkenalkan pertama kali dengan diluncurkannya *GPRS* (*general packet radio system*). *GPRS* merupakan teknologi *overlay* yang disisipkan diatas jaringan *GSM* untuk menangani komunikasi data pada jaringan dengan kata lain dengan menggunakan *handset GPRS*. Komunikasi data tetap berlangsung diatas jaringan *GSM*. Dengan *GSM* masih menangani komunikasi suara dan *transfer* data diatas *GPRS.*

Pengembangan teknologi *GPRS* ditas *GSM* dapat dilakukan secara efektif tanpa menghilangkan infrastruktur lama, yaitu dengan penambahan beberapa *hardware* dan *upgrade software* baru pada terminal dan *server GSM*. Kecepatan *transfer* data *GPRS* dapat mencapai hingga 160 kbps. Teknologi *GPRS* memiliki 3 fitur keunggulan yaitu :

1. *Always online*

   GPRS menghilangkan mekanisme *dial* ke pengguna pada saat ingin mengakses data sehingga dikatakan *GPRS* selalu *online* karena *transfer* data dapat dikirim berupa paket data dan tidak bergantung waktu koneksi.
2. *an upgrade to existing network* (*GSM* dan *TDMA*)

   Adopsi sistem *GPRS* tidak perlu menghilangkan sistem lama karena *GPRS* dijalankan diatas infrastruktur yang sudah ada.
3. *an internal part of edge and WCDMA*.

   GPRS merupakan inti dari mekanisme pengiriman paket data untuk teknologi *3G* selanjutnya.

2.2 SMS (*Short Message Service*)

SMS (*Short Message Service*) pertama kali diperkenalkan pada tahun 1991 di Eropa, ketika teknologi *wireless* pertama kali muncul dan sistem *GSM* (*Global System For Mobile Communication*) telah memiliki layanan tersebut. Dengan *SMS* terminal *mobile station* (*MS*) dapat menerima dan mengirimkan pesan. Pesan singkat sebanyak 16 karakter *alphanumeric*, pentransmisian pesan secara *connectionless* dengan menggunakan kanal pensinyalan.

Pesan dapat dikirim langsung dari *mobile station* (*MS*) ke *mobile station* (*MS*) lainnya tanpa melewati operator, sehingga pesan yang dikirimkan dapat sewaktu-waktu dikirim walaupun *mobile station* (*MS*) yang dituju dalam keadaan tidak aktif atau berada diluar jangkauan. Dengan kemampuan *stread forward* yang dimiliki *SMS* maka pesan tersebut untuk sementara disimpan dalam jaringan sampai *mobile station* (*MS*) yang dituju kembali ke dalam layanan area pelayanan atau siap menerima pesan.

Prinsip kerja *SMS* (*Short Message Service*) adalah setiap jaringan mempunyai *service center* (*SC*). *Service center* ini berfungsi menyimpan dan meneruskan pesan dari pengirim ke pelanggan yang dituju. *SMS* menjadi *interface* antara *public land mobile network* (*PLMN*) *GSM* dengan berbagai sistem lainnya seperti *email*, *fax* atau suatu *content provider*. *Service center* terhubung Ke *PLMN* atau *MSC.*

Berdasarkan fungsinya dalam pengiriman *SMS,* fungsi *MSC* dapat dibedakan menjadi dua yaitu*:*

1. *Gateway MSC for SMS* (*GSM – GSMC*)

   Yaitu fungsi dari *MSC* yang mampu menerima pesan dari *service center* kemudian mencari informasi rutin ke *HLR* selanjutnya mengirim pesan ke *VSMC* dimana pelanggan tersebut berada.

2. *Internet working MSC for SMS* (*SMS-IWSMC*)

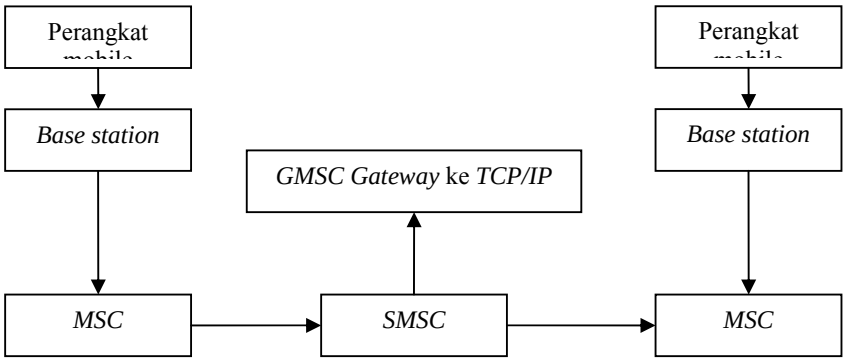
   Yaitu dari *MSC* yang mampu mengirim pesan dari *PLMN* dan memeriksanya ke *service center*.

2.2.1 Arsitektur Jaringan *SMS*

Untuk mengimplementasi layanan *SMS*, operator menyediakan apa yang di *select* sebagai *SMS center* (*SMSC*). Secara fisik *SMSC* terwujud sebuah *personal computer* (*PC*) bisa mempunyai *interkonektivitas* dengan jaringan *GSM*.

Dari gambar arsitektur dasar *SMS* dibawah dapat dilihat *SMSC* memiliki *interkonektivitas* dengan *short messaging entity* (*SME*) yang berupa jaringan

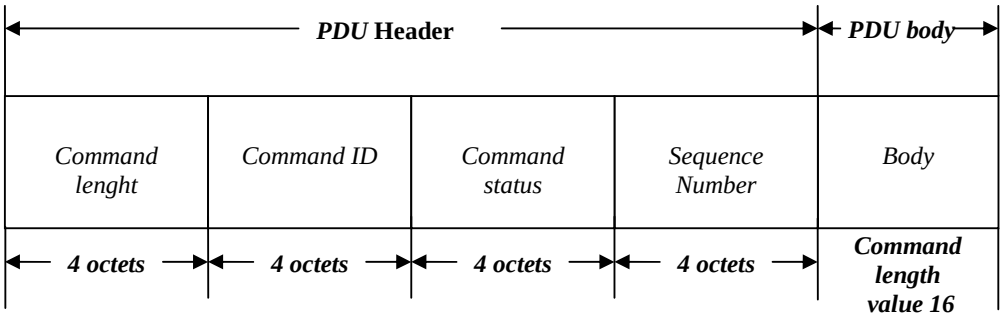
Email, *Web* dan *Voice E-Mail*. *SMSC* inilah yang akan melakukan manajemen pesan *SMS*, baik untuk pengiriman, pengaturan antrian *SMS* dan penerimaan *SMS*.



Gambar 2.1 Arsitektur *SMS*

2.3 PDU (*Protocol Data Unit*)

Protokol data unit merupakan satuan paket data yang disatukan pada lapisan aplikasi dalam *short message peer to peer protocol* (*SMPP*). *SMPP* merupakan sebuah *protocol* standar industri yang disesuaikan dalam pertukaran *short message* antara *external short messaging entity*. Secara umum format *PDU* dalam *SMPP* adalah sebagi berikut :



	SMPP PDU File	*Size(object)*	*Type*
PDU HEADER	*Command lenght*	4	*Integer*
	Command ID	4	*Integer*
	Command status	4	*Integer*
	Sequence Number	4	*Integer*
Body	*Standart parameters*	*var*	*mixed*
	TLV Parameteres	*var*	*mixed*

Gambar 2.2 Format Umum *PDU*

Enam belas oktet pertama disebut *PDU header* merupakan bagian yang harus selalu ada dalam setiap *PDU*. *PDU Body* merupakan bagian yang bersifat *optional* (bisa ada atau tidak bergantung jenis *PDU* nya). *PDU Body* terdiri dari parameter standar dan parameter tambahan yang disebut parameter *TLV* (*tag lenght value*) berikut penjelasan masing-masing bagian tersebut.

1. *Command length*

Menyatakan panjang yang sesungguhnya dari *PDU* meliputi *PDU Header* dan *PDU Body*, termasuk didalamnya *octets* dari *Command length* posisi *Command length* terletak pada awal atau *field* pertama dari setiap *PDU*.

2. *Command ID*

Merupakan parameter yang mengindikasi jenis operasi dari *protocol SMPP* yang sedang dilakukan. *Command ID* dikodekan dalam empat *octet* dengan tipe data *integer* dan urutan pengiriman menganut sistem *Big Endian* (*MSB* terlebih dahulu) nilai parameternya adalah berkisar antara 0x00000000 hingga 0x00000IFF untuk *Command ID* yang termasuk kategori *request*, untuk kategori respon nilainya berkisar antara 0x00000000 hingga 0x800000IFF. Berikut ini daftar beberapa *Command ID* dan nilainya.

Tabel 2.1 Beberapa *Command ID* dan Nilainya

Command ID	Nilai
Bind_receiver	0x00000001
Bind_transmitter	0x00000002
Query_sm	0x00000003
Submit_sm	0x00000004
Deliver_sm	0x00000005

3. *Command status*

Merupakan parameter yang menunjukan status operasi dalam protokol *SMPP* dan hanya relevan untuk *PDU* kategori *respon*, untuk *PDU* kategori *request* parameter ini diabaikan.

4. *Sequence Number*

Merupakan parameter yang digunakan untuk menandai sebuah *PDU* secara unik agar dapat dibedakan dengan lainnya.

Berikut ini *PDU* yang dibutuh kan untuk mengirim *sms* ke *sms center* :

a. Nomor *sms center*.

   Terdiri atas tiga sub *header*

   1. Jumlah pasangan *hexadecimal sms center* adalah bilangan *hexa*.
   2. Kode Nasional / Kode Internasional

      Kode nasional : 81

      Kode internasional : 91
   3. No *sms center* nya sendiri adalah pasangan *hexa* yang saling dipertukarkan dengan urutan tertentu jika terdiri satu angka *hexa* yang tidak memiliki pasangan, untuk angka tersebut dipasangkan dengan huruf F didepannya.

   Sebagai contoh penulis menggunakan *simcard* simpati produk Telkomsel dengan nomor *sms center*_nya 08110000 kode *PDU* nya adalah sebagai berikut :

      1. Cara pertama : menggunakan kode nasional
         1. Sms *center* 0811000000.
         2. Terdapat 06 pasang.
         3. 81 (kode nasional) ada 1 pasang.
         4. 06-11-00-00-00

      Untuk kode *PDU* nya adalah 06818011000000.

      2. Cara kedua : menggunakan kode internasional
         1. *Sms center* 62811000000.
         2. Terdapat 07 pasang.
         3. 91 (kode internasional) ada 1 pasang.
         4. 26-18-01-00-00-F0

      Untuk kode *PDU* nya adalah 0791218010000F0.

b. Tipe *sms*

   Untuk *send* tipe *SMS* adalah 1 dalam *hexa* 01

c. Nomor referensi *SMS*

   nomor referensi ini dibiarkan dalam 0 *hexa* nya adalah 00, ponsel secara otomatis akan memberi nilai nomor referensi.

d. Nomor ponsel penerima

Terdiri atas tiga bagian sub *header* yaitu :

1. Jumlah bilangan desimal ponsel yang dituju dalam bilangan *hexa*.
2. Kode Nasional atau Kode Internasional kode sub *header* nasional adalah 81 dan kode sub *header* internasional adalah 91.
3. Nomor ponsel yang dituju dalam pasangan hexa yang saling dipertukarkan menurut aturan tertentu.

Sebagai contoh untuk nomor ponsel yang dituju 6281321272326 maka dapat ditulis sebagai berikut :

1. Cara pertama : menggunakan kode nasional.
    1. 08 13 21 27 23 26.
    2. Terdapat 12 angka *hexa* nya dalah OC.
    3. 81 (kode nasional).
    4. 80-31-12-72-32-62.

    Berikut kode *PDU* nya adalah : 06818103112723262.
2. Cara kedua : menggunakan kode internasional.
    1. 62 81 32 12 72 32 6.
    2. Terdapat 13 angka *hexa* nya dalah OD.
    3. 91 (kode internasional).
    4. 2618232212732F6.

    Berikut kode *PDU* nya adalah : 0D91261823212732F6.

e. Bentuk *sms*

00 dikirim sebagai *sms.*

01 dikirim sebagi telek.

f. Skema *encoding* data I/O dengan dua skema.

1. Skema 7 *bit,* ditandai dengan angka 00.
2. Skema 8 *bit*, ditandai dengan angka lebih besar 0 (dalam *hexa*).

g. Jangka waktu sebelum.

Jika bagian ini diloncat berarti *sms expired* tidak dibatasi, sedangkan jika di isi dengan bilangan *integer* yang kemudian diubah menjadi bilangan *hexa,* bilangan tersebut memakai jumlah waktu *validitas sms* tersebut.

Tabel 2.2 *Validitas* Waktu *SMS*

Integer (INT)	Jangka waktu *validitas*
0-143	(INT + 1) x 5 menit berarti 5 sampai dengan 12 jam
144-167	12 jam + (INT - 143) x 30 menit
168-196	(INT-166) x 1 hari
197-295	(INT-192) x 1 minggu

h. Isi *sms*

Terdiri dari dua sub *header* yaitu :

1. Panjang isi (jumlah huruf dalam *sms*)

   Misal kata *Hello* terdiri dari 5 huruf.

2. Isi berupa pasangan bilangan *hexa* pada tabel dibawah ini skema 7 *bit* dapat kita lihat :

Tabel 2.3 Skema 7 *bit* Sebagian Karakter

				b7	0	0	0	0	1	1	1	1
				b6	0	0	1	1	0	0	1	1
				b5	0	1	0	1	0	1	0	1
b4	b3	b2	b1		0	1	2	3	4	5	6	7
0	0	0	0	0	@	Δ	SP	0	-	P	-	p
0	0	0	1	1	-	-	!	1	A	Q	a	q
0	0	1	0	2	$	Φ	“	2	B	R	b	r
0	0	1	1	3	-	Γ	#	3	C	S	c	s
0	1	0	0	4	-	-		4	D	T	d	t
0	1	0	1	5	-	-	%	5	E	U	-	-
0	1	1	0	6	-	-	&	6	F	V	-	-
0	1	1	1	7	-	-		7	G	W	-	-
1	0	0	0	8	-	-	)	8	H	X	-	-
1	0	0	1	9	-	-	(	9	I	Y	-	-
1	0	1	0	10	-	-	*	:	J	Z	-	-
1	0	1	1	11	-	-	+	;	K	-	k	-
1	1	0	0	12	-	-	,	<	L	-	l	-
1	1	0	1	13	-	-	-	=	M	-	m	-
1	1	1	0	14	-	-	.	>	N	-	n	-
1	1	1	1	15	-	-	/	?	O	-	o	-

Sebagai contoh untuk kata *HELLO*.

Langkah pertama adalah mengubah menjadi kode 7 *bit*, kemudian langkah selanjutnya adalah mengubah ke dalam kode 8 *bit*.

Langkah pertama :

bit	7	1
H	110	1000
E	110	0101
L	110	1100
L	110	1100
O	110	1111

Langkah kedua :

	E	8
H	1110	1000
	3	2
E	0011	0010
	9	B
L	1001	1011
	F	D
L	1111	1101
	0	6
O	0000	110

Maka kata *Hello* konversinya adalah : E8239BFD06

Setelah selesai langkah a sampai dengan h, maka tinggal menggabungkan sub *header*-sub *header* tersebut. Sehingga didapat *PDU* unutk *send sms* lengkapnya sebagai berikut :

07912618010000F0 01 00 0d91261823212732F6 00 00 E83293FD06.

PDU ini akan diterima oleh *sms center* dan kemudian mengirimkanyan ke nomor tujuan.

Header PDU untuk *sms* terima dari s*ms center* hampir sama dengan *PDU send*, seperti dibawah ini :

1. No *sms*.
2. tipe *sms*, untuk *sms* terima tipenya adalah 04.
3. No *handphone* pengirim.
4. Bentuk *sms*.
5. Skema *encoding*.
6. Tanggal dan waktu *sms*.
7. Batas waktu *validasi*.
8. Isi *sms*.

2.4 SMS Gateway

SMS Gateway adalah elemen yang memberikan layanan koneksi terhadap aplikasi atau elemen lain yang akan menggirimkan pesan. SMS gateway tersebut bisa jadi telah ada dalam *message center* (SMSC) sehingga bukan merupakan elemen terpisah dari *message center*.

SMS Gateway (SMS-GMSC) adalah suatu MSC yang mampu menerima suatu pesan SMS dari satu SMSC, *Home Location Register* (HLR) untuk melalukkan routing informasi, dan mengirimkan SMS ke tujuan, MSC dari stasiun penerima. SMS interworking MSC (SMS-IWMSC) adalah satu MSC yang mampu menerima SMS dari mobile network dan mengirim sesuai SMSC.

Aplikasi ini berupa SMS Server yang berfungsi untuk menampung semua informasi SMS, baik incoming SMS maupun outgoing SMS (two-way) tapi tidak bisa menggunakan sender ID alphanumeric. Untuk menjalankan aplikasi ini dibutuhkan satu atau lebih modem GSM atau bisa juga dengan ponsel GSM yang mempunyai modem dan mensupport AT command. Cukup dengan menambahkan keyword (bisa dibuat sendiri) ke dalam tabel maka SMS akan masuk ke dalam folder-folder tertentu atau menjawab SMS yang masuk sesuai dengan keyword-nya. Dengan aplikasi SMS gateway banyak hal bisa dilakukan seperti polling, kuis, registrasi keanggotaan/membership, pengingat janji (reminder), dan lain-lain masih banyak lagi. Pengembangannya bisa luas sekali tergantung kebutuhuan dan tersedianya database.

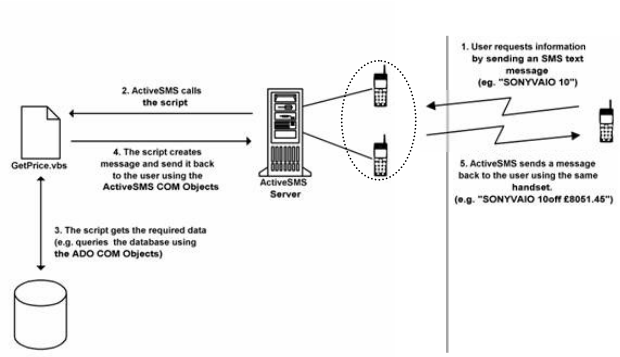
SMS Gateway haruslah dirancang sebagai system/aplikasi yang handal sesuai kebutuhannya. Beberapa fitur SMS Gateway adalah sebagai berikut:

1. Parameterized Keyword & Autoresponder
2. Web based application & web management
3. Multi user (bisa digunakan secara workgroup/intranet)
4. Push SMS (mengirim SMS baik operator GSM maupun CDMA)
5. Pull SMS (mereply SMS baik dari operator GSM maupun CDMA)
6. Grouping SMS

7. Support multi modem (GSM & handphone CDMA)
8. Detail out-going & incoming SMS
9. Quota SMS per user, dll.

2.4.1 Multi Modem Gateway

Multi modem sistem merupakan aplikasi yang membutuhkan dua atau lebih modem GSM atau bisa juga dengan ponsel GSM yang mempunyai modem dan mensupport AT command, aplikasi ini diperlukan dengan alasan tertentu, seperti aplikasi yang memiliki transaksi lebih banyak, untuk antisipasi kesibukan jaringan dan lain sebagainya. Dengan multi modem *client* dapat mengakses aplikasi dengan lebih banyak jalur pilihan. Selain itu kinerja system akan lebih cepat. Gambar 2.3 menunjukkan contoh dari aplikasi sistem multi modem.



Gambar 2.3 Multi Modem Sistem

2.5 Pengertian *Database*

Database adalah sekumpulan dari data-data yang berhubungan satu sama yang lainnya yang dipergunakan untuk pencarian suatu data tertentu pada saat *query* dijalankan.Sebuah *database* dirancang, dibuat serta ditempati oleh data dengan tujuan tertentu. Pada sistem *database* relasional, tabel – tabel dari *database* saling berhubungan satu sama lainnya. Dan sebuah tabel *database* akan selalu memiliki *attribute names*,*relation names*,dan *tuples*.

Basis data dapat didefinisikan dalam sejumlah sudut pandang, seperti :

a. Himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasikan sedemikian rupa agar dapat dimanfaatkan kembali dengan cepat dan mudah.

b. Kumpulan data yang saling berhubungan yang tersimpan aecara bersama sedemikian rupa dan tanpa pengulangan (redudansi) yang tidak perlu, untuk memenuhi berbagai kebutuhan.
c. Kumpulan file atau tabel atau arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronis (Fathansyah, CV Informatika 1999).
d. Melalui penyusunan dan pengelolaan basis data akan didapatkan sejumlah tujuan berikut: (Fathansyah, CV Informatika 1999)
    1. Kecepatan dan kemudahan (Speed)

       Pemanfaatan basis data memungkinkan menyimpan data atau melakukan perubahan/ manipulasi data dan menampilkan kembali data tersebut lebih cepat dan mudah.
    2. Efisiensi Ruang Penyimpanan (Space)

       Karena keterkaitan erat antar kelompok data dalam sebuah basis data, maka redudansi data pasti akan selalu ada. Banyaknya redudansi ini menyebabkan pemborosan ruang penyimpanan. Melalui pemanfaatan basis data, efisiensi penggunaan ruang penyimpanan akan dapat dilakukan.
    3. Keakuratan (Accuracy)

       Pemanfaatan pengkodean atau pembentukan relasi antar data bersama dengan penerapan aturan/ batasan (constraint) tipe data, domain data, keunikan data, dan sebagainya yang secara ketat dapat diterapkan dalam sebuah basis data, sangat berguna untuk menekan ketidakakuratan pemasukan/ penyimpanan data.
    4. Kelengkapan (Completeness)

       Lengkap atau tidaknya data yang di kelola dalam sebuah basis data bersifat relatif (baik terhadap kebutuhan pemakai maupun terhadap waktu).
    5. Keamanan (Security)

       Untuk sistem yang besar, aspek keamanan diperlukan dengan menentukan siapa-siapa saja (pemakai) yang boleh menggunakan basis data.

Dari pengertian diatas maka dapat disimpulkan bahwa *database* memiliki beberapa kriteria penting, yaitu :

a. Bersifat data *oriented* dan bukan program *oriented*.
b. Dapat digunakan oleh beberapa program aplikasi tanpa perlu mengubah basis datanya.
c. Dapat berkembang dengan mudah, baik itu volume maupun strukturnya.
d. Dapat memenuhi kebutuhan sistem-sistem baru secara mudah.
e. Dapat digunakan dengan cara yang berbeda.
f. Kerangkapan data (Data *Redudancy*) minimal

2.5.1 File dan Perancangan *Database*

Dalam perancangan *database* terbagi dalam beberapa tahap sebagai berikut:

1. Pemodelan data, proses pembangunan *database* untuk menentukan tabel mana yang dibutuhkan suatu program aplikasi.
2. Pemantapan sasaran kedalam proses operasional, untuk meyakinkan sasaran dari perancangan *database* diperlukan pemantapan yang membuat proses operasional lebih terperinci kedalam suatu *database* sederhana.
3. Penentuan objek *database* yang dibutuhkan.
4. Memeriksa keberadaan *database* sesuai dengan tipenya, pemeriksaan ini berguna pada saat pembuatan normalisasi *database* pada suatu sistem.
5. Membuat bingkai kerja (*Framework*) *database*, pada tahap ini pembuatan bingkai *database* terbagi kedalam beberapa bagian yaitu membuat nama field dengan tipenya dan pengefisienan ukuran *field*.
6. Pengorganisasian tabel dengan tipenya, proses ini membagi-bagi peletakan suatu *field* dalam *database*.
7. Pengambaran tabel berelasi, agar lebih mudah dalam perancangan *database* diperlukan suatu tabel sehingga dalam menentukan mana master data atau data tambahan.

8. Normalisasi *database*, normalisasi data merupakan eleminasi dari *database* yang berlebihan (penggunaan data lebih dari satu tabel) dari tiap tabel dalam suatu *database*.

2.6 My SQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa Inggris: *database management system*) atau DBMS yang *multithread*, *multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL merupakan database yang handal dan telah teruji untuk menangani berbagai aplikasi berbasis web. Database MySQL ini memiliki kemampuan dan kecepatan database yang handal. Ketangguhan sistem Database MySQL untuk menangani data dalam jumlah besar ini menjadi pertimbangan untuk digunakan dalam sistem informasi pemetaan potensi dan Implementasi masa sekarang. MySQL juga terbukti sangat kompatibel dengan PHP dan mendukung pembuatan aplikasi networking.

Tidak sama dengan proyek-proyek seperti Apache, dimana perangkat lunak dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia MySQL AB, dimana memegang hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB adalah: David Axmark, Allan Larsson, dan Michael "Monty" Widenius.

Terdapat beberapa API tersedia yang memungkinkan aplikasi-aplikasi komputer yang ditulis dalam berbagai bahasa pemrograman untuk dapat mengakses basis data MySQL antara lain: bahasa pemrograman C, C++, C#, bahasa pemrograman Eiffel, bahasa pemrograman Smalltalk, bahasa pemrograman Java, bahasa pemrograman Lisp, Perl, PHP, bahasa pemrograman Python, Ruby, REALbasic dan Tcl. Sebuah antarmuka ODBC memanggil MyODBC yang memungkinkan setiap bahasa pemrograman yang mendukung ODBC untuk berkomunikasi dengan basis data MySQL. Kebanyakan kode sumber MySQL dalam ANSI C.

MySQL sangat populer dalam aplikasi web seperti MediaWiki (perangkat lunak yang dipakai Wikipedia dan proyek-proyek sejenis) dan PHP-Nuke dan berfungsi sebagai komponen basis data dalam LAMP. Popularitas sebagai aplikasi web dikarenakan kedekatannya dengan popularitas PHP, sehingga seringkali disebut sebagai *Dynamic Duo*.

Untuk melakukan administrasi dalam basis data MySQL, dapat menggunakan modul yang sudah termasuk yaitu *command-line* (perintah: mysql dan mysqladmin). Juga dapat diunduh dari situs MySQL yaitu sebuah modul berbasis grafik (*GUI*): *MySQL Administrator* dan *MySQL Query Browser*. Selain itu terdapat juga sebuah perangkat lunak gratis untuk administrasi basis data MySQL berbasis web yang sangat populer yaitu phpMyAdmin. Untuk perangkat lunak untuk administrasi basis data MySQL yang dijual secara komersial antara lain: MySQL front, Navicat dan EMS SQL Manager for MySQL.

2.7 *Personal Computer*

Istilah komputer (*computer*) berasal dari bahasa Latin *computare* yang berarti menghitung. Komputer mempunyai arti yang sangat luas dan berbeda menurut penggunaanya. Berikut ini definisi komputer yang didapat dari beberapa buku komputer. Menurut buku ***computer annual*** (Robert H Blissmer) Komputer adalah suatu alat elektronik yang mampu melakukan bebreapa tugas sebagai berikut :

a. Menerima *input*
b. Memproses *input* tadi sesuai dengan programnya.
c. Menyimpan perintah-perintah dari dari hasil pengolahan.
d. Menyediakan *output* dalam bentuk informasi.

Menurut buku *computer today* (Donald H Sanders) komputer adalah sistem elektronik untuk memanipulasi data yang cepat dan tepat serta dirancang dan diorganisasikan supaya secara otomatis menerima dan menyimpan data *input*, memprosesnya dan menghasilkan *output* dibawah pengawasan suatu langkah-langkah instruksi program yang tersimpan di memori (*storage memory*).

Menurut buku *computer organization* (V.C Hamacher, Z.G Vranesic, S.G Zaky) komputer adalah mesin penghitung elektronik yang cepat dan dapat menerima informasi *input* digital, memprosesnya sesuai dengan suatu program yang tersimpan di memorinya (*storage memory*) dan menghasilkan *output* informasi. Menurut buku *introduction to computer* (Gordon B Davis) komputer adalah tipe khusus alat penghitung yang mempunyai sifat tertentu yang pasti.

Dari beberapa definisi diatas, dapat disimpulkan bahwa komputer adalah :

a. Alat elektronik
b. Dapat menerima *input* data
c. Dapat mengolah data
d. Dapat memberikan informasi
e. Menggunakan suatu program yang tersimpan di *storage memory*.
f. Dapat menyimpan program dan hasil pengolahan.
g. Bekerja secara otomatis.

Sedangkan yang disebut dengan program adalah kumpulan instruksi-instruksi atau perintah terperinci yang sudah dipersiapkan supaya komputer dapat melakukan fungsinya dengan cara yang sudah ditentukan.

2.8 *Visual Basic*

Visual Basic adalah sebuah bahasa pemrograman generasi ke4 (fourth generation programming language) yang diciptakan oleh Microsoft, yang dikembangkan dari bahasa *BASIC* (*Beginner's All Purpose Symbolic Intruction Code*) yang dikembangkan di era 1950-an. *visual basic* merupakan salah satu *development tool* yaitu alat bantu untuk membuat berbagai macam program komputer. Khususnya yang menggunakn sistem operasi *windows*. *Visual Basic* merupakan salah satu bahasa pemrograman komputer yang mendukung objek *OOP* (*object oriented programing)*. Beberapa karakteristik obyek tidak dapat dilakukan pada *Visual Basic*, seperti *Inheritance* tidak dapat dilakukan pada *class module*. *Polymorphism* secara terbatas bisa dilakukan dengan mendeklarasikan *class module* yang memiliki Interface tertentu.

Dalam lingkungan *Window's User-interface* sangat memegang peranan penting, karena dalam pemakaian aplikasi yang kita buat, pemakai senantiasa berinteraksi dengan *user-interface* tanpa menyadari bahwa dibelakangya berjalan dengan intruksi-intruksi program yang mendukung tampilan dan proses yang dilakukan. Pada pemrograman visual, pengembangan aplikasi dimulai dengan pembentukan *user-interface,* kemudian mengatur property dari obyek-obyek yang digunakan dalam *user-interface,* dan baru dilakukan penulisan kode program untuk menangani kejadian-kejadian (event). Tahap pengembangan aplikasi demikian dikenal dengan istilah pengembangan aplikasi dengan pendekatan *Bottom-Up.*

2.9 Rekayasa Perangkat Lunak

Perangkat lunak merupakan suatu program yang berisi sejumlah intruksi yang dibuat untuk melakukan sejumlah proses dengan tujuan sebagai alat pengendalian kegunaan sistem atau untuk menyelesaikan kesulitan suatu aplikasi tertentu. Dalam membangun suatu perangkat lunak selalu timbul berbagai macam kesulitan. Kesulitan yang terdapat dalam pembangunan perangkat lunak dapat diatasi dengan cara mengenali permasalahan yang ada. Tapi solusi ini harus dapat memberikan bantuan yang berguna bagi pembangunan perangkat lunak dan meningkatkan kualitas perangkat lunak dan akhirnya penerapan perangkat lunak pada perangkat keras.

Definisi menurut Fritz Bauer *"Rekayasa Perangkat Lunak adalah pengembangan dan penggunaan prinsip pengembangan suara untuk memperoleh perangkat lunak secara ekonomis yang reliable dan bekerja secara efisien pada mesin nyata".*

Menurut IEEE [IEE93] mengembangkan definisi yang lebih komprehensif rekayasa, rekayasa perangkat lunak yaitu :

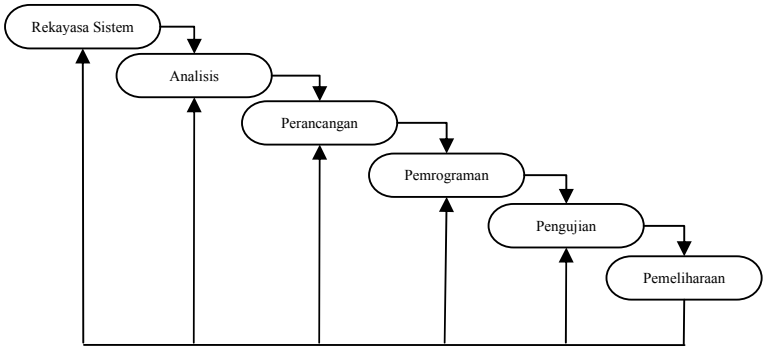
1. Aplikasi dari sebuah pendekatan *kuantifiabel*, disiplin, dan sistematis kepada pengembangan, operasi, dan pemeliharaan perangkat lunak.
2. Studi tentang pendekatan-pendekatan seperti pada nomor satu.

2.9.1 Rekayasa Sistem Berorientasi Objek

Rekayasa adalah bentuk upaya manusia untuk mencari atau merancang produk yang berkualitas. Produk dapat berupa sistem atau perangkat lunak yang sedang atau akan dibangun dengan suatu model pengembangan.

Orientasi objek adalah cara pandang bukan sekedar algoritma yang diimplementasikan pada bahasa pemrograman berorientasi objek, cara berpikir orientasi objek adalah segala sesuatu dipandang sebagai objek. Objek dapat berupa konsep, abstraksi atau sesuatu dengan batas-batas tegas dan mempunyai arti untuk persoalan yang ditangani. Objek mempunyai identitas dan dapat dibedakan. Ada dua kegunaan objek, yaitu untuk meningkatkan pemahaman dunia nyata dan menyediakan dasar pengetahuan praktis untuk implementasi komputer.

Model yang digunakan dalam proses perancangan perangkat lunak masih seperti model yang digunakan dalam metodologi konvensional, yaitu dapat menggunakan model *life cycle* atau *waterfall*. Dalam model *life cycle*, terdapat lima tahapan proses, yaitu tahap rekayasa sistem, analisis, perancangan, pemrograman, pengujian, pemeliharaan. Dengan menggunakan *waterfall*, memungkinkan adanya pengembangan aplikasi di kemudian hari, sehingga *software* yang dibangun dapat bertahan dalam jangka waktu yang panjang.



Gambar 2.4 Tahapan model *waterfall*

1. Rekayasa sistem

Merupakan kegiatan untuk mempelajari dan menganalisis kebutuhan sistem, data yang mengalir dan perangkat penunjang seperti *software* dan *hardware* yang digunakan untuk membangun sistem.

2. Analisis

Menentukan apakah kegiatan dari rekayasa sistem dapat diimplementasikan menjadi sebuah sistem atau tidak dan menentukan prosedur-prosedur yang bekerja. Adapun prosedur - prosedur tersebut meliputi masukan (*input*), pemrosesan (*processing*), dan keluaran (*output*).

3. Perancangan

Tahap desain objek menggunakan desain berorientasi objek, secara spesifik mempunyai arti bahwa kita mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya. Konsep berorientasi objek dapat dilakukan untuk semua sistem bahasa yang digunakan, yaitu UML (*Unified Modelling Language*), dan menggunakan beberapa diagram dari UML tersebut seperti :

a. *Use Case Diagram* : Diagram ini menjelaskan manfaat sistem jika dilihat menurut pandangan orang yang berada diluar sistem (*actor*) dan menunjukan fungsionalitas suatu sistem atau kelas dan bagaimanan sistem berinteraksi dengan dunia luar.

b. *Activity diagram* : Diagram ini memodelkan alur kerja (*workflow*) sebuah proses bisnis dan urutan aktifitas dalam suatu proses.

c. *Sequence diagram* : Diagram ini menunjukkan kolaborasi antara objek dan menunjukan bagaimana urutan operasi berlangsung, pesan apa yang dikirim, dan kapan pesan dikirimkan.

d. *Class Diagram* : Diagram ini menggambarkan struktur kelas-kelas dari suatu sistem dan memperlihatkan hubungan antar kelas dan penjelasan detail tiap-tiap kelas didalam model desain dari suatu sistem.

4. Pengkodean

Merupakan kegiatan yang mengimplementasikan hasil dari perancangan perangkat lunak ke dalam kode program agar pengguna (*user*) dapat memahami sistem yang sedang dibangun.

5. Pengujian

Memfokuskan pada fungsi internal, fungsi eksternal dari perangkat lunak dan mencari segala kemungkinan kesalahan, memeriksa apakah input data sesuai dengan hasil yang diinginkan setelah proses.

6. Pemeliharaan

Merupakan kegiatan untuk memelihara (*maintenance*) perangkat lunak yang telah dibangun, pemeliharaan tersebut dilakukan agar keutuhan program dapat terjaga.

2.10 Definisi UML (*Unified Modeling Language*)

UML *(Unified Modelling Language*) 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.

UML (*Unified Modelling Language*) adalah bahasa pemodelan untuk menspesifikasikan, memvisualisasikan, membangun dan mendokumentasikan artifak-artifak dari sistem.

1. Didalam *system intensive process*, metode diterapkan sebagai proses untuk menurunkan atau mengevolusikan sistem.
2. Sebagai bahasa, UML digunakan untuk komunikasi yaitu alat untuk menangkap pengetahuan mengenai satu subyek dan mengekspresikan pengetahuan (sintaks) yang memperdulikan subyek untuk maksud komunikasi. Subyek adalah sistem yang dibahas.
3. Sebagai bahasa pemodelan, UML focus pada pemahaman subyek melalui formulasi model dari subyek. Model memuat pengetahuan pada subyek, dan aplikasi dari pengetahuan ini berkaitan dengan intelejensia.
4. Berkaitan dengan unifikasi.UML memadukan praktek rekayasa terbaik sistem informasi dan industri, meliputi beragam tipe sistem (perangkat

lunak dan non perangkat lunak), domain (bisnis, perangkat lunak) dan proses siklus hidup.

5. begitu diterapkan untuk menspesifikasikan sistem, UML dapat digunakan untuk mengkomunikasikan "apa" yang diperlukan dari sistem dan "bagaimana" sistem dapat direalisasikan.
6. Begitu diterapkan untuk menvisualisasikan sistem,UML dapat digunakan untuk menjelaskan sistem secara visual sebelum direalisasikan.
7. Begitu diterapkan untuk membangun sistem, UML dapat digunakan untuk memandu realisasi sistem serupa dengan "blueprint".
8. Begitu diterapkan untuk mendokumentasikan sistem, UML dapat digunakan untuk menangkap pengetahuan mengenai sistem pada seluruh siklus hidup.

UML bukanlah :

1. Bahasa pemrograman visual, tapi bahasa pemodelan visual.
2. Spesifikasi kakas, tapi spesifikasi bahasa pemodelan.
3. Proses, tapi yang memungkinkan proses-proses.

2.10.1 Tujuan UML

Tujuan utama perancangan UML adalah :

1. Menyediakan bahasa pemodelan visual yang ekspresif dan siap pakai untuk mengembangkan dan pertukaran model-model yang berarti.
2. Menyediakan mekanisme perluasan dan spesifikasi untuk memperluas konsep-konsep inti.
3. Mendukung spesifikasi independen bahasa pemrograman dan proses pengembangan tertentu.
4. Menyediakan basis formal untk pemahaman bahasa pemodelan.
5. Mendorong pertumbuhan pasar kakas berorientasi objek.
6. Mendukung konsep-konsep pengembangan level lebih tinggi seperti komponen, kolaborasi, *framework* dan *pattern.*

2.10.2 Diagram dan Teknik Pemodelan UML

Diagram mengemukakan banyak hal, penggunaan notasi yang terdefinisi baik dan ekspresif adalah penting pada proses pengembangan perangkat lunak, yaitu :

1. Notasi standar memungkinkan pengembang mendeskripsikan skenario atau rumusan arsitektur dan kemudian mengkomunikasikan secara tidak ambigu.
2. Notasi yang bagus membebaskan otak untk berkonsentrasi pada masalah-masalah yang lebih lanjut.
3. Notasi yang baik memungkinkan mengeleminasi keperluan pemeriksaan konsistensi dan kebenaran keputusan keputusan dengan menggunakan tool terotomatisasi.

2.10.3 Diagram struktur

Diagram ini untuk memvisualisasi, menspesifikasikan, membangun dan mendokumentasikan aspek statik dari sistem. Diagram struktur di UML terdiri dari

1. Diagram kelas (*Class diagram*)
2. Diagram objek (*Objek diagram*)
3. Diagram komponen (*Component diagram*)
4. Diagram deployment (*Deployment diagram*)

2.10.4 Diagram perilaku

Diagram ini untuk memvisualisasi, menspesifikasi, membangun dan mendokumentasikan aspek dinamis dari sistem. Diagram perilaku di UML terdiri dari :

1. Diagram *use-case* (*Use case diagram*)
2. Diagram sekuen (*Sequen diagram*)
3. Diagram kolaborasi (*Collaboration diagram*)
4. Diagram *state*chart (*Statechart diagram*)
5. Diagram aktivitas (*Activity Diagram*)

1. Diagram kelas (*Class diagram*)

Diagram ini menunjukkan sekumpulan kelas, *interface* dan kolaborasi dan keterhubungannya. Diagram kelas ditujukan untuk pandangan statik terhadap sistem.

2. Diagram objek (*Object diagram*)

Diagram ini menunjukkan sekumpulan objek dan keterhubungannya. Diagram ini menunjukkan potongan statik dari instan–instan yang ada di diagram kelas. Diagram ini untuk memperlihatkan satu *prototype* atau kasus tertentu yang mungkin terjadi. Diagram obyek menyediakan notasi grafis formal guna memodelkan objek, kelas, dan saling keterhubungan. Diagram objek berguna untuk abstract modeling dan perancangan program–program sesungguhnya.

Pada pendekatan ini, bentukan dasar dari sistem perangkat lunak adalah objek atau kelas. Kelas adalah deskripsi dari objek–objek yang common. Setiap objek mempunyai identitas, state dan perilaku.

3. Diagram use-case (*Use case diagram*)

Diagram ini menunjukkan sekumpulan kasus fungsional dan *actor* (jenis kelas khusus) dan keterhubungannya.

4. Diagram sekuen (*Sequence diagram*)

Diagram ini menunjukkan interaksi yang terjadi antar objek. Diagram ini merupakan pandangan dinamis terhadap sistem. Diagram ini menekankan pada basis keberurutan waktu dari pesan pesan yang terjadi.

5. Diagram kolaborasi (*Collaboration diagram*)

Diagram ini juga merupakan diagram interaksi. Diagram ini menekankan pada organisasi struktur dari objek-objek yang mengirim dan menerima pesan.

6. Diagram *state*chart (*Statechart diagram*)

Diagram ini adalah *state-machine* diagram, berisi state, transisi, kejadian dan aktivitas. Statechart merupakan pandangan dinamis dari sistem. Diagram ini

penting dalam memodelkan perilaku antarmuka, kelas, kolaborasi dan menekankan pada urutan kejadian. Penting untuk sistem reaktif yang dipicu kejadian di dunia nyata.

7. Diagram aktivitas (*Activity diagram*)

Diagram ini untuk menunjukkan aliran aktivitas di sistem. Diagram ini adalah pandangan dinamis terhadap sistem. Diagram ini penting untuk memodelkan fungsi sistem dan menekankan pada aliran kendali di antara objek–objek.

8. Diagram komponen (*Component diagram*)

Diagram ini menunjukkan organisasi dan kebergantungan di antara sekumpulan komponen. Diagram ini merupakan pandangan statik terhadap implementasi sistem.

9. Diagram deployment (*Deployment diagram*)

Diagram ini menunjukkan konfigurasi pemrosesan saat jalan dan komponen komponen yang terdapat didalamnya. Diagram ini merupakan pandangan static dari arsitektur. Pilihan model dan diagram yang digunakan dipengaruhi oleh bagaimana persoalan ditangani dan bagaimana solusi dibentuk. Abstraksi, focus pada relevan sambil mengabaikan rincian rincian yang tidak relevan merupakan kuncinya. Karena itu, setiap sistem kompleks perlu didekati melalui sekumpulan pandangan model yang hampir *independent*. Tidak ada satu pandangan tunggal yang memadai untuk sistem kompleks.

2.10.5 Notasi dalam UML (*Unified Modeling Language*)

UML menyediakan beberapa notasi dan artifak standar yang bisa digunakan sebagai alat komunikasi bagi para pelaku dalam proses analisis dan desain. Artifak di dalam UML didefinisikan sebagai informasi dalam berbagai bentuk yang digunakan atau dihasilkan dalam proses pengembangan perangkat lunak.Yang perlu diperhatikan untuk menjaga konsisten antar artifak, selama proses analisis dan desain adalah bahwa setiap perubahan yang terjadi pada satu artifak harus juga dilakukan pada artifak sebelumnya.

1. *Actor*

Actor adalah segala sesuatu yang berinteraksi dengan sistem aplikasi komputer. Jadi actor ini bisa berupa orang, perangkat keras, atau mungkin objek lain dalam sistem yang sama. Biasanya yang dilakukan actor adalah memberikan informasi pada sistem dan atau memerintahkan sistem untuk melakukan sesuatu.

2. *Class*

Class merupakan pembentuk utama dari sistem berorientasi objek karena class menunjukkan kumpulan objek yang memiliki atribut dan operasi yang sama. *Class* digunakan untuk mengabstraksikan elemen-elemen dari sistem yang sedang dibangun. *Class* bisa untuk merepresentasikan baik perangkat lunak maupun perangkat keras, baik konsep maupun benda nyata. Notasi *class* berbentuk persegi panjang berisi 3 bagian. Persegi panjang paling atas untuk nama class, persegi panjang paling bawah untuk operasi dan persegi panjang tengah untuk atribut. Atribut digunakan untuk menyimpan informasi, nama atrribut menggunakan kata benda yang bisa dengan jelas merepresentasikan informasi yang disimpan didalamnya. Operasi menunjukkan sesuatu yang bisa dilakukan oleh objek dan menggunakan kata kerja.

3. *Interface*

Interface merupakan kumpulan operasi tanpa implementasi dari suatu class. Implementasi operasi dalam *interface* dijabarkan oleh operasi dalam *class*. Oleh karena itu keberadaan *interface* selalu disertai oleh *class* yang mengimplementasikan operasinya. *Interface* ini merupakan salah satu cara mewujudkan prinsip enkapsulasi dalam objek.

4. *Use Case*

Use case menjelaskan urutan kegiatan yang dilakukan *actor* dan sistem untuk mencapai suatu tujuan tertentu walaupun menjelaskan kegiatan. Namun *use case* hanya menjelaskan "apa" yang dilakukan oleh *actor* dan sistem, bukan "bagaimana" *actor* dan sistem melakukan kegiatan tersebut.

5. *Interaction*

Interaction digunakan untuk menunjukkan baik aliran pesan atau informasi antar obyek maupun hubungan antar obyek. Biasanya *Interaction* ini dilengkapi juga dengan teks bernama *operation signature* yang tersusun dari nama operasi, parameter yang dikirim dan tipe parameter yang dikembalikan.

6. *Package*

Package adalah kontainer atau wadah konseptual yang digunakan untuk mengelompokkan elemen-elemen dari sistem yang sedang di bangun, sehingga bisa dibuat model yang lebih sederhana. Tujuannya adalah untuk mempermudah penglihatan (*visibility*) dari model yang sedang dibangun.

7. *Note*

Note digunakan untuk memberikan keterangan dan komentar tambahan dari satu elemen sehingga bisa langsung terlampir dalam model. *Note* ini bisa ditampilka ke semua elemen notasi yang lain.

8. *Dependency*

Dependency merupakan relasi yang menunjukkan bahwa perubahan pada salah satu elemen memberi pengaruh pada elemen lain. Elemen yang ada dibagian tanda panah adalah elemen yang tergantung pada elemen yang ada di bagian tanpa tanda panah. Terdapat 2 *stereotype* dari *dependency*, yaitu *Include* dan *Extend.* Individu menunjukkan bahwa suatu kegiatan dari elemen (yang ada digaris tanpa panah) memicu eksekusi bagian dari elemen lain (yang ada digaris dengan panah), misalnya untuk notasi A → B operasi yang ada di *class* A Memicu dieksekusinya operasi yang ada di class B. *Extended* menunjukkan bahwa suatu bagian dari elemen di garis tanpa bisa disisipkan ke dalam elemen yang ada di garis dengan panah, misalnya untuk notasi A→ B suatu fungsi dari *use case* A bisa disisipkan ke dalam *use case* B atau dengan kata lain A optional untuk B. Kedua *stereotype* ini direpresentasikan dengan menambahkan *text include* atau *extend* di notasi *dependency*.

9. *Boundary Class*

Boundary class adalah *class* yang menghubungkan *user* dengan sistem. Oleh karena itu sering disebut juga sebagai *user interface class.*

10. *Control class*

Control class adalah *class* yang mengkoordinasi aktivitas dalam sistem. Class ini menghubungkan *boundary class* dengan *entity class.*

11. *Entity Class*

Entity Class adalah *class* yang menghubungkan dengan data atau informasi yang digunakan oleh sistem. *Entity class* ini adalah *class* yang menyimpan dan mengelola data dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*).