BAB II

LANDASAN TEORI

2.1. Ruang Lingkup Sistem *Inventory* dan Penembakan *Voucher* Elektronik MKIOS

Sistem *Inventory* dan penembakan *voucher* elektronik MKIOS pada beberapa perusahaan *Sub Dealer* (SD) Telkomsel ini telah berjalan cukup lama dan sempat mengalami beberapa perubahan bentuk sistem. Pada awal mula hadirnya *voucher* elektronik MKIOS pihak perusahaan sangat kurang memperhatikan sektor ini sehingga sistem yang berjalan dapat disimpulkan bergerak secara tradisional dengan menggunakan buku besar sebagai media penyimpanan data dan tidak ada pola yang terstruktur dalam proses manajemen dan *Inventory* nya. Karena dianggap sudah tidak dapat menunjang lagi, sistem pada fase yang pertama diganti dengan sistem semi manual atau fase kedua, dikatakan semi manual karena pada fase ini sistem manajemen sudah menggunakan bantuan media komputer dengan memakai software *Microsoft Office* sederhana. Namun pada bagian *Inventory* proses pencatatan masih mengunakan media buku atau kartu stok yang disalin secara manual. Sistem ini digunakan sampai dengan sekarang karena memang belum ada sistem baru yang bisa berjalan lebih baik. Namun bukan berarti sistem pada fase ini tidak ada kekurangan, banyak sekali peluang terjadinya kesalahan-kesalahan dalam proses pelaksanaannya. Berdasarkan pengamatan bobot dari proses yang dijalankan secara manual adalah 60% dan 40% nya dengan bantuan komputer.

Pembangunan perangkat lunak otomatisasi sistem manajemen dan *Inventory* voucher elektronik MKIOS ini merupakan perpaduan proses manual dan proses berbantuan komputer pada fase kedua yang akan di integrasikan menjadi satu kesatuan yang saling berhubungan.

Beberapa gambaran proses manual yang akan diotomatiskan adalah :

1. **Pembuatan faktur penjualan**

   Pembuatan faktur penjualan adalah proses pembuatan bukti pembelian voucher elektronik yang dibuat oleh perusahaan untuk *Reseller*.

2. **Data pelaporan**

   Data pelaporan adalah laporan dari keseluruhan tahapan yang isinya akan disesuaikan dengan setiap level *user* dalam perusahaan.

3. **Sistem Inventory**

   Sistem Inventory ini merupakan wadah dari *history* keluar masuknya produk dan informasi sisa produk dalam hal ini *voucher* elektronik.

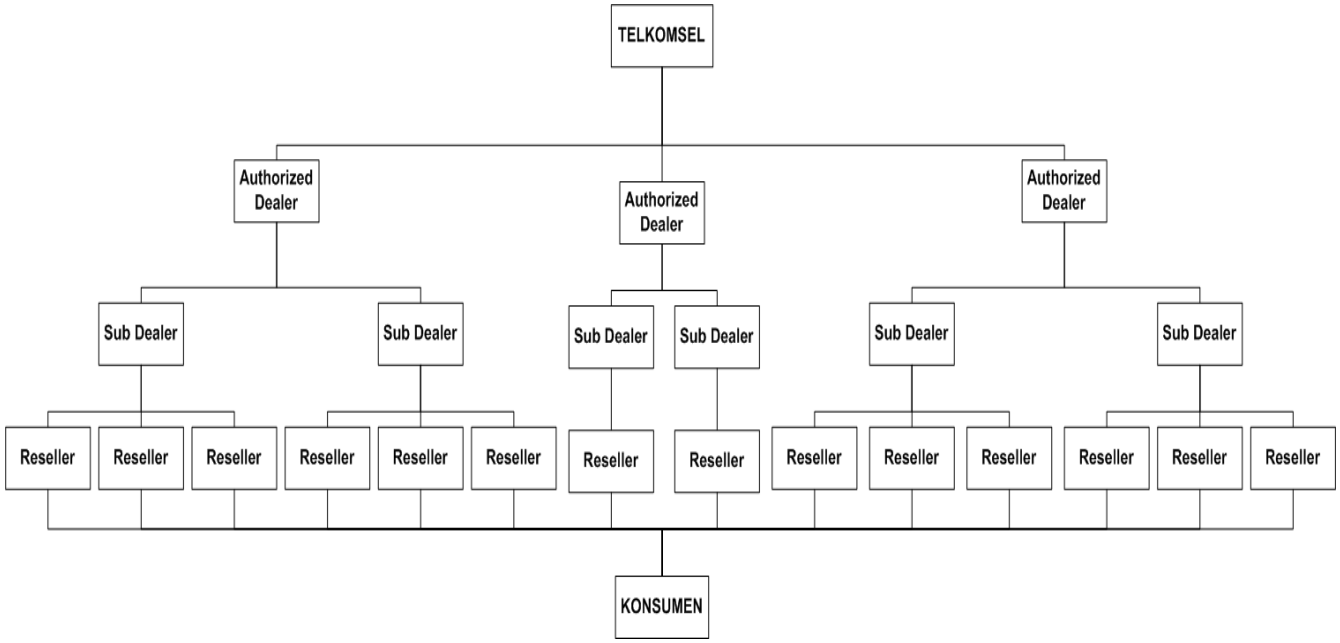
4. **Sistem Penembakan MKIOS**

   Sistem Penembakan MKIOS adalah proses pengiriman *voucher* elektronik yang telah dipesan kepada reseller yang bersangkutan.

   \

2.2. Hirarki Pendistribusian Produk MKIOS

Dalam hirarkinya pendistribusian produk MKIOS memiliki struktur sebagai berikut :



Gambar 2.1. Struktur Distribusi Produk MKIOS

Dari Gambar 2.1 dapat dijelaskan bahwa proses pendistribusian produk MKIOS diawali dari TELKOMSEL sebagai *provider* yang mendistribusikan produk MKIOS kepada level *Authorized Dealer* di bawahnya. Selanjutnya produk MKIOS tersebut didistribusikan oleh *Authorized Dealer* kepada beberapa level *Sub Dealer* dimana pembagiannya diatur sesuai dengan kebijakan perusahaan *Authorized Dealer* yang terkait. Setelah beberapa *Sub Dealer* menerima alokasi produk MKIOS dari *Authorized Dealer* kemudian produk MKIOS yang ada didistribusikan kepada beberapa *Reseller* yang terdaftar untuk dijual kepada konsumen.

2.3. SMS (*Short Message Service*)

2.3.1. Sejarah SMS

SMS *(Short Message Service)* adalah salah satu fasilitas dari teknologi GSM yang memungkinkan mengirim dan menerima pesan singkat berupa text dari MS

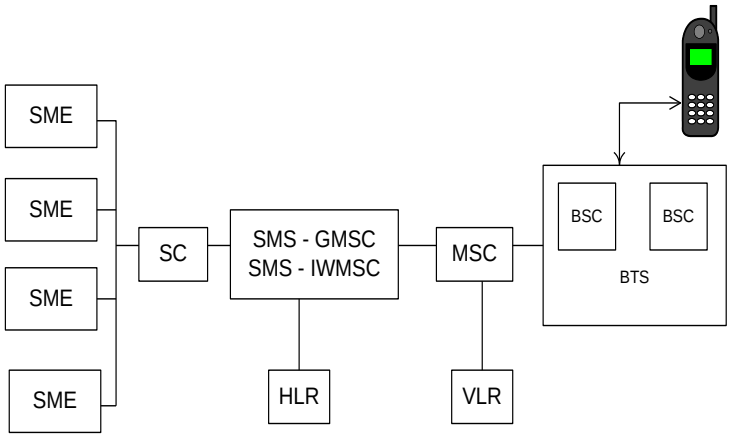
(Mobile Station). Layanan SMS juga memungkinkan pengiriman pesan dalam bentuk *alphanumeric,* layanan SMS ini banyak diaplikasikan pada sistem komunikasi tanpa kabel (*wireless).* Teknologi *wireless* dipelopori dari kawasan Eropa yang diawali pada kebutuhan bersama terhadap satu sistem jaringan baru yang dapat menjadi standar jaringan yang berlaku dan dapat diterapkan di seluruh kawasan Eropa. Dalam sistem baru juga harus terdapat kemampuan yang dapat mengantisipasi mobilitas pengguna serta kemampuan melayani lebih banyak pengguna untuk menampung penambahan jumlah pelanggan baru.

Karena hal ini tidak dapat dilakukan dengan mempertahankan sistem analog, maka kemudian diputuskan untuk merombak sistem dan menggantinya dengan sistem digital. Standard baru diperkenalkan dengan nama GSM *(Global Standard for Mobile Communications).* GSM pada awalnya adalah kepanjangan dari *Groupe Speciale Mobile,* sebuah badan gabungan dari para ahli yang melakukan studi bersama untuk menciptakan standard GSM tersebut.

Pada bulan Desember 1992, dilakukan pengiriman pesan menggunakan SMS dari sebuah PC *(Personal Computer)* ke telepon *mobile* (bergerak) dalam jaringan GSM milik Vodafone Inggris, kemudian merambah ke benua Amerika yang dipelopori oleh beberapa operator komunikasi bergerak berbasis digital seperti BellSouth Mobility, PrimeCo dan operator lainnya. Teknologi yang digunakan dari pengiriman SMS yaitu *Store* and *forward service,* jadi SMS yang di kirim akan simpan sementara di *server* SMS *center* kemudian dialihkan ke nomor tujuan.

2.3.2. Arsitektur dan Elemen Jaringan SMS

Layanan SMS dibangun dari berbagai entitas yang saling terkait dan mempunyai fungsi dan tugas masing-masing. Tidak ada satupun dalam sistem SMS yang dapat bekerja secara parsial. Entitas dalam jaringan SMS ini disebut juga elemen SMS. Pada Gambar 2.1 diperlihatkan arsitektur SMS dengan beberapa elemen-elemen yang saling terkait :



Gambar 2.2 Arsitektur dan Elemen Jaringan SMS

Elemen-elemen dasar pada jaringan SMS :

a. *SME (Short Message Entity)*, merupakan tempat penyimpanan dan pengiriman pesan yang akan dikirimkan ke MS tertentu.
b. *SC (Service Centre)*, bertugas untuk menerima pesan dari SME dan melakukan *forwarding* ke alamat MS yang dituju.
c. *SMS-GMSC (Short Message Service-Gateway SMC*), melakukan penerimaan pesan dari SC dan memeriksa parameter yang ada. Selain itu GMSC juga mencari alamat MS yang dituju dangan bantuan HLR, dan mengirimkannya kembali ke MSC yang dimaksud.

d. SMS-IWMSC *(Short Message Service-Interworking MSC*), berperan dalam SMS *Message Originating*, yaitu menerima pesan dari MSC

e. HLR *(Home Location Register)* merupakan sebuah *database* yang digunakan sebagai tempat penyimpanan permanen data dan profil pelanggan. Bila diminta oleh SMSC, maka HLR dapat memberikan informasi *routing* dari pelanggan tertentu. HLR juga dapat memberikan informasi status tujuan apakah aktif atau tidak.

f. VLR *(Visitor Location Register)* merupakan sebuah *database* tempat menyimpan informasi sementara yang berisi data pelanggan dari sebuah HLR yang *roaming* pada HLR lain.

g. MSC merupakan sebuah sistem yang melakukan fungsi *switching* dan mengontrol panggilan telepon dalam sebuah jaringan komunikasi bergerak. MSC inilah yang akan mengirimkan sebuah SMS ke suatu tujuan tertentu melalui *base station* yang sesuai.

h. BSS *(Base Station System)* Merupakan kesatuan sistem yang bertanggung jawab mengatur transmisi sinyal elektromagnetik untuk membawa data dari MSC ke perangkat telepon bergerak. *Base Station* terdiri dari BSC *(Base Station Controler)* dan BTS *(Base Tranceiver Station)* dan juga dikenal dengan nama *cell cite* atau sederhananya *cell*. Sebuah BSC biasanya menangani satu atau lebih BTS dan bertanggung jawab menangani pelanggan saat berpindah dari satu *cell* ke *cell* lainnya.

i. *Mobile Device* merupakan perangkat yang mempunyai kemampuan mengirimkan dan menerima SMS, biasanya berupa telepon seluler dengan teknologi digital. Akan tetapi, saat ini jenis terminal berkembang sesuai aplikasi dan kebutuhan seperti POS, laptop dan PDA *(Personal Digital Assistant)*.

2.3.3. Layanan Aplikasi SMS

Layanan aplikasi SMS merupakan sebuah layanan yang bersifat *none real time* dimana sebuah SMS dapat di *submit* ke suatu tujuan, tidak peduli apakah tujuan tersebut aktif atau tidak. Bila terdeteksi bahwa tujuan tidak aktif, maka sistem akan menunda pengiriman ke tujuan hingga tujuan aktif kembali. Pada dasarnya sistem SMS akan menjamin *delivery* dari suatu SMS hingga sampai tujuan. Kegagalan pengiriman yang bersifat sementara seperti tujuan tidak aktif akan selalu teridentifikasi sehingga pengiriman ulang SMS akan selalu dilakukan kecuali apabila diberlakukan aturan bahwa SMS yang telah melampaui batas waktu tertentu harus dihapus dan dinyatakan gagal kirim. Berdasarkan mekanisme distribusi pesan SMS diperoleh aplikasi SMS, terdapat 4 macam mekanisme pengantaran pesan yaitu :

a. *Pull*, yaitu pesan yang dikirimkan ke pengguna berdasarkan permintaan pengguna.
b. *Push Event based*, yaitu pesan yang diaktivasi oleh aplikasi berdasarkan kejadian yang berlangsung
c. *Push Schedule*, yaitu pesan yang diaktivasi oleh aplikasi berdasarkan waktu yang telah terjadwal.

d. *Push Personal Profile*, yaitu pesan yang diaktivasi oleh aplikasi berdasarkan *profile* dan *preference* dari pengguna

2.3.4. Prinsip Kerja SMS

Prinsip Kerja dari SMS diantaranya adalah sebagai berikut :

a. Prinsip kerja dari SMS ini adalah bahwa setiap jaringan mempunyai suatu SC *(service center).* Pesan tidak langsung dikirimkan ke tempat tujuan melainkan di simpan terlebih dahulu di SC menjadi *interface* antara PLNM *(public land mobile network).*
b. Transmisi SMS dapat terjadi meskipun MS *(mobile station)* sedang melakukan komunikasi dengan MS yang lain. Hal ini di mungkinkan karena kanal radio untuk transmisi *voice* telah ditentukan selama durasi pemanggilan sedangkan SMS merambat pada kanal radio dengan memanfaatkan jalur *signal*
c. Pengiriman SMS yang menggunakan kanal *signal* memiliki dua tipe, yaitu :
    1. SMS *Point To point* yaitu menyediakan mekanisme untuk mengirimkan pesan hanya dari satu MS ke MS tertentu, berupa pesan pendek ke dan dari piranti bergerak
    2. SMS *broadcast (Point to multipoint)* pengiriman SMS ke beberapa MS sekaligus
d. Pesan yang tidak terkirim, akan diberikan informasi report yang menyatakan pesan SMS gagal terkirim

e. Jika ponsel tidak aktif untuk sms akan tetap masuk dan di simpan di SMS dengan waktu tertentu, jika ponsel aktif sebelum batas waktu maka SMS akan di kirimkan.

2.4. PDU *(Protocol Data Unit)*

Langkah-langkah untuk menentukan kode PDU yang dibutuhkan untuk mengirimkan SMS ke SMS *Center* :

a. Nomor SMS *Center*

Nomor SMS *Center* terdiri dari tiga *subheader* yaitu:

1. Jumlah pasangan heksa desimal SMS *Center* dalam bilangan heksa.
2. Kode nasional dan kode internasional, nomor kode nasional yang digunakan adalah 81 sedangkan nomor kode internasional adalah 91.
3. Nomor SMS *Center*, dalam pasangan bilangan heksa yang saling dipertukarkan dengan aturan tertentu, jika tersisa satu angka heksa yang tidak memiliki pasangan maka angka tersebut dipasangkan dengan huruf F didepannya.

Contoh konversi nomor ponsel kedalam format kode PDU dengan menggunakan :

1. Kode Nasional

SMS *Center* menggunakan nomor 0856000000 maka :

a. Terdapat 6 pasang angka heksa desimal.

b. Kode nasional 81, ada 1 pasang

c. 80-56-00-00-00

Maka kode PDU yang didapat : 06818056000000

2. Kode Internasional

SMS *Center* menggunakan nomor 62811000000, maka:

a. Terdapat 7 pasang angka heksa desimal

b. Kode internasional 91, ada 1 pasang

c. 26-18-01-00-00-F0

Maka kode PDU yang didapat : 07912618010000F0

Pada tabel 2.1 diperlihatkan beberapa nomor SMS *Center* operator selular di Indonesia :

Tabel 2.1 *Nomor SMS-Center (a) Operator Selular dan Kode PDU (Cara I) dan (b) Operator Selular dan Kode PDU (Cara II)*

No	Operator Selular	SMS-*Center* No	Kode PDU
1	Telkomsel	0811000000	06818011000000
2	Satelindo	0816125	0581806121F5
3	Excelcom	0818445009	06818081440590
4	Indosat-M3	0855000000	06818055000000

Keterangan : Operator selular dan kode PDU (Cara I)

No	Operator Selular	SMS-*Center* No	Kode PDU
1	Telkomsel	6281000000	07912618010000F0
2	Satelindo	62816125	059126181652
3	Excelcom	62818445009	07912618485400F9
4	Indosat-M3	62855000000	07912658050000F0

Keterangan : Operator selular dan kode PDU (Cara II)

b. Tipe SMS, untuk pengiriman atau *send* tipe SMS adalah 1 dalam heksa 01

c. Nomor referensi SMS, nomor ini dibiarkan dulu 0 atau dalam heksa 00, ponsel secara otomatis akan memberi nilai nomor referensinya.

d. Nomor ponsel penerima, terdiri atas tiga bagian *subheader* yaitu:

1. Jumlah bilangan desimal ponsel yang dituju dalam bilangan heksa
2. Kode nasional atau kode internasional, kode *subheader* nasional adalah 81 dan internasional adalah 91.
3. Nomor ponsel yang dituju dalam pasangan heksa yang saling dipertukarkan menurut aturan tertentu.

Contoh untuk nomor ponsel yang dituju adalah 085221960301, maka konversi kedalam heksa desimal yaitu dengan cara seperti yang dijabarkan berikut.

1. Kode Nasional :
   a. terdapat 12 angka, berarti dalam heksa adalah 0C
   b. kode nasional adalah 81
   c. 80-25-12-69-03-01

   Maka kode *subheader* PDU yang didapat adalah 0C81802512690301
2. Kode Internasional
   a. 62 81 32 12 17 32 6
   b. Terdapat 13 angka, berarti dalam heksa adalah 0D
   c. Kode internasional adalah 91
   d. 261823212732F6

   Maka kode *subheader* PDU yang didapat adalah 0D91261823212732F6
3. Bentuk SMS

   00 : dikirim sebagai SMS

   01 : dikirim sebagai Telekomunikasi
4. Skema *encoding* data *Input* dan *Output* (I/O)

Terdapat dua skema yaitu :

Skema 7 bit, ditandai dengan angka 00

Skema 8 bit, ditandai lebih besar dari nol (dalam heksa)

5. Jangka Waktu atau validasi sebelum SMS *expired* atau kadaluarsa, jika bagian ini terlewati itu berarti berlakunya SMS tidak dibatasi, sedangkan jika diisi dengan suatu bilangan integer yang kemudian diubah ke dalam bilangan heksa, maka bilangan tersebut mewakili jumlah waktu validasi SMS tersebut. Pada tabel 2.2 diperlihatkan validitas waktu SMS.

Tabel 2.2 *Validitas Waktu SMS*

Integer (INT)	Jangka Waktu Validasi
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
197-255	(INT-192) x 1 minggu

6. Isi SMS

Terdiri dari dua subheader yaitu:

a. Panjang Isi (Jumlah huruf dalam SMS), misal untuk kata “hello” terdiri dari 5 huruf. Isi berupa pasangan bilangan heksa desimal, skema 7 bit dapat dilihat pada Tabel 2.3 dibawah ini.

Tabel 2.3 *Konversi Heksadesimal*

				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		*¿*	*#*	*3*	*C*	*S*	*C*	*s*
0	*1*	*0*	*0*	4				*4*	*D*	*T*	*D*	*t*
0	*1*	*0*	*1*	5			*%*	*5*	*E*	*U*	*E*	*u*
0	*1*	*1*	*0*	6			*&*	*6*	*F*	*V*	*F*	*v*
0	*1*	*1*	*1*	7			*.*	*7*	*G*	*W*	*G*	*w*
1	*0*	*0*	*0*	8			*(*	*8*	*H*	*X*	*H*	*x*
1	*0*	*0*	*1*	9			*)*	*9*	*I*	*Y*	*I*	*y*
1	*0*	*1*	*0*	10	*LF*		*	*:*	*J*	*Z*	*J*	*z*
1	*0*	*1*	*1*	11			+	*;*	*K*		*K*	
1	*1*	*0*	*0*	12			*‘*	<	*L*		*L*	
1	*1*	*0*	*1*	13	*CR*		*_*	=	*M*		*M*	
1	*1*	*1*	*0*	14				>	*N*		*N*	
1	*1*	*1*	*1*	15			*/*	*?*	*O*		*O*	

Ada 2 langkah yang harus kita lakukan untuk mengkonversikan isi SMS , yaitu :

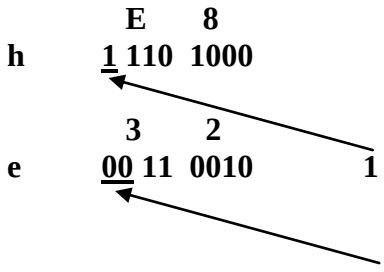
1. Langkah pertama yaitu mengubah menjadi kode 7 bit

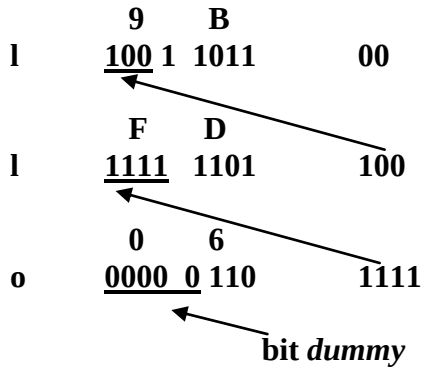
   Contoh untuk kata “hello”

Langkah pertama	**Bit**	**7**	**1**
	h	110	1000
	e	110	0101
	l	110	1100
	l	110	1100
	o	110	1111

2. Langkah kedua yaitu mengubah kode 7 bit menjadi 8 bit yang diwakili oleh pasangan heksa.

   Langkah kedua





Sehingga hasil konversi kata “hello” ke bilangan heksa adalah **E8329BFD06.**

Oleh karena total 7 bit x 5 huruf = 35 bit, sedangkan yang kita perlukan untuk mengubah ke 8 bit adalah 8 bit x 5 huruf = 40 bit, maka diperlukan 5 bit *dummy* yang diisi bilangan 0.

Setelah masing-masing *header* dan *subheader* untuk mengirim pesan pesan dipecahkan, maka *header-header* diatas digabung menjadi sebuah PDU lengkap.

Seperti contoh untuk mengirim kata “Hello”ke MS nomor 628129573337 lewat SMS *Center* Exelcom tanpa membatasi jangka waktu valid, maka PDU lengkapnya adalah :

07912618485400F901000C9126I892753373000005E8329BFD06

7. Tanggal dan waktu SMS di-*stamp* di SMS *Center* diwakili oleh 12 bilangan heksa (6 pasang) yang berarti YY/MM/DD hh:mm:ss

   Contoh :

   207022512308 02/07/22 12:32:08 22 Juli2002 15:23:08 WIB

2.5. AT- *Command*

AT-*Command* atau *Attention Command* yaitu perintah AT (Hayes AT-*Command)* yang digunakan untuk berkomunikasi dengan terminal (modem)

melalui gerbang serial pada komputer. Dengan penggunaan perintah AT, dapat diketahui atau dibaca kondisi dari terminal, seperti mengetahui kondisi sinyal, kondisi baterai, mengirim pesan, membaca pesan, menambah item pada daftar telepon, dan sebagainya. Beberapa jenis ponsel memiliki *extended* AT *Command* yang bisa digunakan untuk mengambil informasi jenis, model hp, nomor *Internasional Mobile Station Equipment Identity* (IMEI) , SIM *Subscriber Identification Number* (IMSI), status baterai, kekuatan sinyal, nama operator, lokasi dan *cell* ID. Pada tabel 2.4 berikut diperlihatkan beberapa jenis perintah Hayes yang berhubungan dengan penanganan pesan-pesan AT-Command GSM versi 07.07.

Tabel 2.4. *Perintah AT-Command pada GSM Versi 07.07*

AT-Command	**Singkatan**	**Fungsi**
ATE1	*Activate Command*	Mengaktivasi At-Command 07.07
AT+C.....	*Attention + Command.....*	Awal perintah hayes untuk....
AT+COPS	*Operator Selection*	Memilih operator jaringan
AT+CLCK	*Lock*	Menampilkan penguncian telepon
AT+CBC	*Battery Charge*	Menampilkan level baterei
AT+CSQ	*Signal Quality*	Menampilkan kualitas sinyal
AT+CCLK	*Clock Mode*	Pengaturan Jam
AT+CPBR	*Phone Book Read*	Membaca daftar buku telepon
AT+CPBS	*Phone Book Selection*	Menampilkan isi buku telepon
AT+CCSA	*Service Center Address*	Menampilkan Lokasi *Service Center*
AT+CMGS	*Message Get Send*	Mengirimkan SMS
AT+CMGL	*Message Get List*	Membaca isi *inbox* dan *outbox* SMS
AT+CMGD	*Message Get Delete*	Menghapus isi *SMS*

2.6. Pengertian *Database*

Sebuah sistem *database* meliputi beberapa komponen yang secara bersama-sama memberikan perbedaan tertentu dan karakteristik yang khusus. Berikut ini beberapa definisi ringkas dari *database* :

"Database adalah sistem file komputer yang menggunakan cara perorganisasian file tertentu, dimaksudkan untuk mempercepat pembaruan masing-masing record, serta pembaruan secara serempak atas record terkait, juga untuk mempermudah dan mempercepat akses terhadap seluruh record lewat program-program aplikasi, serta akses yang cepat terhadap data yang tersimpan yang harus digunakan secara bersama-sama untuk dibaca guna penyusunan laporan-laporan rutin atau khusus ataupun untuk penyelidikan." (Azhar Susanto, 2000).

"Database merupakan kumpulan data-data yang tersimpan di dalam media penyimpanan disuatu perusahaan (arti luas) atau di dalam komputer (arti sempit)." (Andri Kristianto, 2004).

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

1. Bersifat data *oriented* dan bukan program *oriented*.
2. Dapat digunakan oleh beberapa program aplikasi tanpa mengubah basis datanya.
3. Dapat berkembang dengan mudah, baik itu volume maupun strukturnya.
4. Dapat memenuhi kebutuhan sistem-sistem baru secara mudah.
5. Dapat digunakan dengan cara yang berbeda.
6. Kerangkapan data (*Data Redudancy*) minimal.

2.6.1. 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 ke dalam 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. Penggambaran 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*.

2.7. Rekayasa Perangkat Lunak

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.

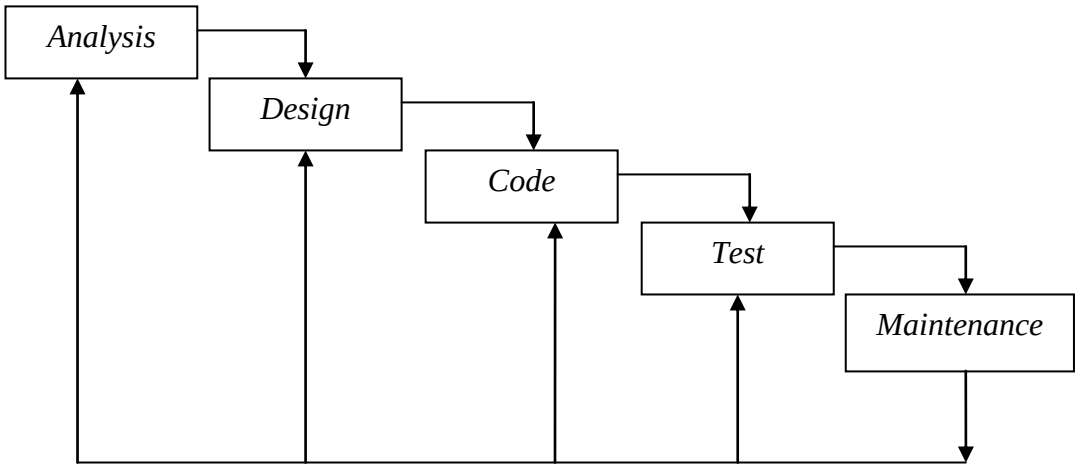
2.8. Rekayasa Sistem Berorientasi Objek

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 2.3 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.

2.9. Definisi Aliran Data Berorientasi Objek

Pemrograman berorientasi objek, atau sering juga disebut dengan OOP (*Object Oriented Programming*), merupakan paradigma baru dalam pengembangan sistem/perangkat lunak. Tidak seperti pendahulunya, yaitu pemrograman dengan teknik terstruktur (*structured programming*) yang sering mengalami kegagalan dalam hal sistem/perangkat lunak yang tidak sesuai dengan kebutuhan dan harapan pengguna (*user's need and expectations*) serta kerap kali tidak dapat memenuhi jadwal yang telah ditentukan sebelumnya. Pemrograman berorientasi objek dengan paradigma barunya, menjanjikan tingkat penggunaan ulang (*reusability*), kinerja (*performance*), serta kehandalan (*reliability*) yang lebih tinggi. (Roger Pressman, 2000).

Saat ini, dunia pemrograman berorientasi objek mengenal UML (*Unified Modelling Language*) sebagai alat utama dalam analisis dan perancangan sistem. UML merupakan alat yang sangat sesuai dengan pemrograman berorientasi objek sebab konsep dasarnya adalah memodelkan kelas-kelas (beserta atribut serta

operasi di dalamnya) bersamaan dengan relasi-relasi yang terjadi antar kelas yang bersangkutan.

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 adalah bahasa grafis untuk mendokumentasikan, menspesifikasikan dan membangun sistem perangkat lunak. UML berorientasi objek, menerapkan banyak level abstraksi, tidak bergantung proses pengembangan, tidak bergantung bahasa dan teknologi, pemaduan beberapa notasi di beragam metodologi, usaha bersama dari banyak pihak, didukung oleh kakas-kakas yang diintegrasikan oleh XML. Standar UML dikelola oleh OMG (*Object Management Group*).

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

1. Di dalam *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 fokus 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.

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 untuk pemahaman bahasa pemodelan.
5. Mendorong pertumbuhan pasar kakas berorientasi objek.
6. Mendukung konsep-konsep pengembangan level lebih tinggi seperti komponen, kolaborasi, *framework* dan *pattern.*

2.10.1. Diagram 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 untuk berkonsentrasi pada masalah-masalah yang lebih lanjut.
3. Notasi yang baik memungkinkan mengeleminasi keperluan pemeriksaan konsistensi dan kebenaran keputusan keputusan dengan menggunakan *tool* terotomatisasi.

2.10.2. 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*)

   Diagram ini menunjukkan sekumpulan kelas, *interface* dan kolaborasi dan keterhubungannya.

2. Diagram objek (*Object diagram*)

   Diagram ini menunjukkan sekumpulan objek dan keterhubungannya.

3. Diagram komponen (*Component diagram*)

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

4. Diagram deployment (*Deployment diagram*)

   Diagram ini menunjukkan konfigurasi pemrosesan saat jalan dan komponen komponen yang terdapat didalamnya.

2.10.3. 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*)

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

2. Diagram Sequen (*Sequence diagram*)

   Diagram ini menunjukkan interaksi yang terjadi antar objek. Diagram ini merupakan pandangan dinamis terhadap sistem.

3. Diagram kolaborasi (*Collaboration diagram*)

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

5. Diagram *statechart* (*Statechart diagram*)

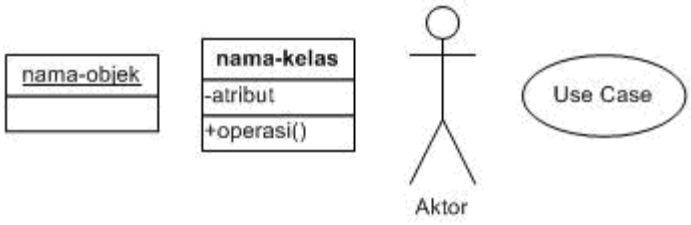
   Diagram ini adalah *state-machine* diagram, berisi state, transisi, kejadian dan aktivitas. Statechart merupakan pandangan dinamis dari sistem.

6. Diagram aktivitas (*Activity Diagram*)

   Diagram ini untuk menunjukkan aliran aktivitas di sistem. Diagram ini adalah pandangan dinamis terhadap sistem.

2.10.4. 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.



Gambar 2.4 Beberapa Notasi UML

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.

3. ***Interface***

   Interface merupakan kumpulan operasi tanpa implementasi dari suatu *class*.

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. ***Intercation***

   Interaction digunakan untuk menunjukkan baik aliran pesan atau informasi antar obyek maupun hubungan antar objek.

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 ditampilkan ke semua elemen notasi yang lain.

8. *Dependency*

Dependency merupakan relasi yang menunjukkan bahwa perubahan pada salah satu elemen memberi pengaruh pada elemen lain.

9. *Boundary Class*

Boundary class adalah *class* yang menghubungkan *user* dengan sistem.

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.

2.11. *Tools* Pembangun

2.11.1. *Microsoft Visual Studio* 2005

Microsoft Visual Studio 2005, yang sebelumnya dikenal dengan *codename* ***Whidbey***, merupakan kumpulan lengkap alat-alat pengembangan untuk membangun aplikasi Web ASP.NET, XML Web Service, aplikasi desktop, dan aplikasi mobile. Di dalam Visual Studio 2005 terdapat beberapa perangkat lunak untuk pengembang (programmer), antara lain Visual Basic 2005 (Visual Basic 8 / Visual Basic . Net 2005), Visual C#, Visual J#, dan Visual C++ yang semuanya menggunakan lingkungan pengembangan terintegrasi atau *Integrated Development Environment* (IDE) yang sama sehingga memungkinkan untuk saling berbagi tool dan fasilitas.

Dalam sejarah perkembangannya, Visual Studio .Net 2002 dirilis dengan berbasis pada Microsoft .NET Framework versi 1.0. Generasi berikutnya adalah Visual Studio .NET 2003, yang menggunakan .NET Framework 1.1. Untuk generasi terbaru, Visual Studio .NET 2005 atau lebih sering disebut Visual Studio 2005 saja, mengandalkan teknologi .NET Framework 2.0.

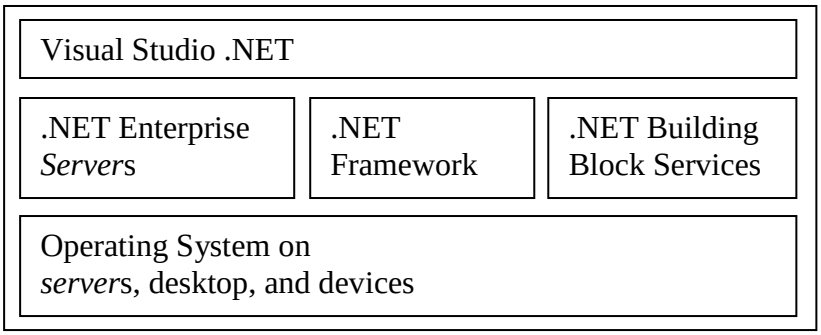
2.11.2. *Platform Microsoft .NET*

Microsoft .NET, yang awalnya disebut *Next Generation Windows Services* (NGWS), adalah suatu platform untuk membangun dan menjalankan generasi penerus aplikasi-aplikasi terdistribusi. Microsoft .NET merupakan framework pengembangan yang menyediakan antarmuka pemrograman baru untuk layanan Windows dan API (*Application Programming Interface*), dan mengintegrasikan sejumlah teknologi Microsoft.

Definisi lain dari Microsoft .NET adalah suatu platform XML (*Extensible Markup Language*) *Web Service* yang memungkinkan kita untuk membangun aplikasi-aplikasi .NET, melakukan interaksi dengan internet, dan menggunakan RAD (*Rapid Application Development*) seperti Visual Studio .NET.

Microsoft .NET merupakan strategi Microsoft untuk menghubungkan sistem, informasi, dan alat (*device*) sehingga orang dapat melakukan komunikasi dan kolaborasi dengan lebih efektif. Teknologi .NET terintegrasi penuh melalui produk-produk Microsoft, dan menyediakan kemampuan untuk mengembangkan solusi menggunakan Web service.

Platform Microsoft .NET terdiri atas lima komponen utama, yang tersusun dalam tiga lapisan (*layer*). Lapisan paling bawah adalah sistem operasi, kemudian lapisan kedua terdiri atas tiga komponen, dan lapisan teratas adalah Visual Studio .NET.



Gambar 2.5. Platform Microsoft .NET

Dalam arsitektur Microsoft .NET, .NET *Framework* berperan sebagai infrastuktur dan merupakan komponen inti. Sementara itu .NET *Enterprise Server* berfungsi untuk menyederhanakan dan mengurangi waktu yang diperlukan pada saat mengembangkan sistem bisnis berskala besar. Di sisi lain, .NET *Building*

Block Services atau .NET *Services* adalah layanan-layanan terdistribusi yang tersedia secara online dan offline, contohnya .NET Passport.

2.11.3. .NET *Framework*

.NET adalah "*platform* perangkat lunak", suatu lingkungan netral bahasa yang digunakan untuk mengembangkan dan membangun aplikasi-aplikasi yang dengan mudah dan aman akan beroperasi di dalamnya. Pada saat aplikasi-aplikasi di-deploy ke komputer tujuan, aplikasi akan dieksekusi di mana saja .NET diimplementasikan. Adapun komponen-komponen yang menyusun *platform* .NET secara bersama disebut .NET *Framework*.

.NET *Framework* adalah model pemrograman *platform* .NET yang digunakan untuk membangun, men-deploy, dan menjalankan XML *Web service* serta aplikasi. .NET *Framework* berisi *library-library class* untuk menyediakan layanan standar yang dapat diintegrasikan ke berbagai sistem komputer. Framework ini mengatur semua aspek eksekusi program, seperti alokasi memori untuk penyimpanan data dan instruksi, eksekusi aplikasi, izin akses aplikasi, dan re-alokasi memori pada *resource* yang tidak diperlukan lagi.

.NET *Framework* memiliki dua komponen utama, yaitu *Common Language Runtime* (CLR) dan *Base Class Library* (BCL). Penjelasan mengenai CLR dan BCL adalah sebagai berikut:

1. *Common Language Runtime* (CLR)

   CLR di .NET *Framework* bertugas untuk mengelola memori, mengeksekusi thread, mengeksekusi kode, verifikasi keamanan kode, melakukan kompilasi, menangani eksepsi lintas bahasa, binding dinamik, dan menyediakan layanan-

layanan sistem lainnya sehingga menjadikan proses pengembangan aplikasi semakin mudah.

Pada saat kode program .NET dikompilasi, keluaran yang dihasilkan bukan merupakan file executable, tetapi berupa kode yang disebut *Microsoft Intermediate Language* (MSIL), dan disimpan di file *assembly*. MSIL mendefinisikan kumpulan instruksi independen terhadap CPU spesifik, dan tugas CLR adalah menerjemahkan kode MSIL ini menjadi kode yang dapat dieksekusi (*executable*). Selanjutnya, kode *executable* tersebut dapat dijalankan di lingkungan apa pun yang mengimplementasikan CLR. Ini merupakan cara agar .NET *Framework* menghasilkan aplikasi yang *portable*.

Proses konversi MSIL ke kode *executable* dilakukan menggunakan kompiler JIT (*Just In Time*), di mana ketika program dijalankan, CLR dengan segera akan mengaktifkan kompiler JIT. Kompiler JIT hanya mengkonversi bagian assembly yang diperlukan saja sehingga menjadikan aplikasi sangat ringan ketika dijalankan. Pada saat kode dikompilasi dengan JIT, CLR akan memeriksa untuk memastikan bahwa kode sesuai. Proses validasi MSIL ini dinamakan dengan verifikasi. Aturan penggunaan CLR untuk verifikasi ini ditetapkan di dalam *Common Language Specification* (CLS) dan *Common Type System* (CTS).

CLR menyediakan kemudahan untuk mendesain komponen dan aplikasi yang objek-objeknya mampu berinteraksi dengan berbagai bahasa. CLR secara otomatis juga menangani layout objek dan mengelola acuan ke objek, dan melepas objek jika sudah tidak digunakan lagi.

2. *Base Class Library* (BCL)

   BCL .NET *Framework* merupakan koleksi tipe-tipe *reusable* yang terintegrasi dengan CLR. BCL menyediakan kumpulan class, *interface*, dan tipe-tipe nilai yang mampu memberi akses ke utilitas spesifik dan berbagai fungsi sistem. BCL memungkinkan kita untuk mengembangkan bermacam aplikasi, mulai dari aplikasi *command-line*, *Graphical User Interface* (GUI) hingga aplikasi *Web Form* dan XML *Web Services*.

Library-library class .NET *Framework* diatur dalam hierarki *namespace*, yakni suatu skema penamaan logikal untuk mengelompokkan tipe-tipe yang saling berkaitan. Dengan demikian, *namespace* memudahkan pemrogram ketika ingin menggunakan library class, serta mencegah terjadinya ambiguitas (nama yang sama) dan konflik penamaan di *class*.

Sebagaimana yang diharapkan dari suatu *library class* berorientasi objek, tipe-tipe .NET *Framework* memungkinkan kita untuk tugas-tugas umum di bidang pemrograman. Termasuk di dalamnya adalah manajemen string, koleksi data, koneksi database, dan pengaksesan file. Selain itu, library-library class juga menyertakan tipe-tipe yang mendukung skenario pengembangan khusus.

Pada dasarnya, .NET *Framework* adalah kumpulan tool, teknologi, dan bahsa yang semuanya bekerja secara bersama-sama di dalam *framework* untuk menyediakan solusi praktis pengembangan aplikasi. Tujuan-tujuan pokok dari .NET *Framework* adalah sebagai berikut:

1. Untuk menyediakan suatu lingkungan pemrograman berorientasi objek yang konsisten, baik itu kode objek yang disimpan dan dieksekusi secara lokal,

dieksekusi secara lokal tetapi terdistribusi melalui internet, atau dieksekusi secara *remote*.

2. Untuk menyediakan suatu lingkungan pengeksekusian kode yang mampu meminimalkan penyebaran perangkat lunak dan terjadinya konflik versi.
3. Untuk menyediakan suatu lingkungan eksekusi kode yang menjamin pengeksekusian kode dengan aman, dan mengurangi performansi pada lingkungan skrip atau interpretasi.
4. Untuk membuat para pengembang memiliki cara kerja yang konsisten ketika bekerja dengan bermacam jenis aplikasi.
5. Membangun seluruh komunikasi pada standar industri untuk memastikan bahwa kode yang berbasis .NET Framework dapat berintegrasi dengan kode lainnya.

2.11.4. Visual Basic.NET 2005

Visual Basic .NET merupakan salah satu bahasa pemrograman yang bisa digunakan untuk membangun aplikasi-aplikasi .NET di *platform* Microsoft .NET. Tidak seperti generasi sebelumnya (Visual Basic 6.0), yang lebih difokuskan untuk pengembangan aplikasi desktop, Visual Basic .NET memungkinkan para pengembang membangun bermacam aplikasi, baik desktop maupun aplikasi web.

Versi pertama Visual Basic .NET diluncurkan dengan label Visual Basic .NET 2002, kemudian disusul dengan Visual Studio .NET 2003. Di Visual Basic .NET 2005, Microsoft menambahkan bermacam kemampuan baru yang berbasis pada .NET *Framework* 2.0. Selain menghilangkan kekurangan-kekurangan di

versi 2002/2003, Visual Basic .NET 2005 juga menawarkan solusi yang lebih praktis dalam mengembangkan aplikasi-aplikasi .NET.

Agar Visual Basic 2005 dapat digunakan dengan baik dan optimal, ada beberapa persyaratan berkaitan spesifikasi perangkat lunak serta perangkat keras yang perlu kita perhatikan. Adapun spesifikasi pokok yang perlu kita perhatikan adalah sebagai berikut:

1. Prosesor berkecepatan 600 MHz (rekomendasi 1 GHz)
2. RAM 192 MB (rekomendasi 256 MB)
3. Kapasitas media penyimpanan sebesar 3,8 GB untuk instalasi penuh
4. Sistem operasi Windows 2000 Service Pack 4 atau Windows XP Service Pack 2 atau Windows *Server* 2003.
5. Display minimum 800 x 600 256 colors (rekomendasi 1024 x 768 high color 16 bit)

2.12. SQL (*Structure Query Language*)

SQL (*Structured Query Language*) adalah sebuah bahasa yang dipergunakan untuk mengakses data dalam basis data relasional basis data relasional. Bahasa ini secara de facto merupakan bahasa standar yang digunakan dalam manajemen basis data relasional. Saat ini hampir semua *server* basis data yang ada mendukung bahasa ini untuk melakukan manajemen datanya.

Sejarah SQL dimulai dari artikel seorang peneliti dari IBM bernama EF Codd yang membahas tentang ide pembuatan basis data relasional pada bulan Juni 1970. Artikel ini juga membahas kemungkinan pembuatan bahasa standar untuk mengakses data dalam basis data tersebut. Bahasa tersebut kemudian diberi nama

SEQUEL (Structured English Query Language). Setelah terbitnya artikel tersebut, IBM mengadakan proyek pembuatan basis data relasional berbasis bahasa SEQUEL. Akan tetapi, karena permasalahan hukum mengenai penamaan SEQUEL, IBM pun mengubahnya menjadi **SQL**. Implementasi basis data relasional dikenal dengan *System/R*. Di akhir tahun 1970-an, muncul perusahaan bernama Oracle yang membuat *server* basis data populer yang bernama sama dengan nama perusahaannya. Dengan naiknya kepopuleran Oracle, maka SQL juga ikut populer sehingga saat ini menjadi standar de facto bahasa dalam manajemen basis data.

2.12.1. Standarisasi

Standarisasi SQL dimulai pada tahun 1986, ditandai dengan dikeluarkannya standar SQL oleh ANSI. Standar ini sering disebut dengan SQL86. Standar tersebut kemudian diperbaiki pada tahun 1989 kemudian diperbaiki lagi pada tahun 1992. Versi terakhir dikenal dengan SQL92. Pada tahun 1999 dikeluarkan standar baru yaitu SQL99 atau disebut juga SQL99, akan tetapi kebanyakan implementasi mereferensi pada SQL92.

Saat ini sebenarnya tidak ada *server* basis data yang 100% mendukung SQL92. Hal ini disebabkan masing-masing *server* memiliki dialek masing-masing.