# BAB II

# LANDASAN TEORI

## 2.1. *Global System for Mobile Communication* (GSM).

*Global system for mobile communication* (GSM) merupakan standar yang diterima secara global untuk komunikasi selular digital. GSM adalah nama group standardisasi yang di mapankan pada tahun 1982 untuk menghasilkan standar telefon bergerak di eropa, digunakan sebagai formula spesifikasi untuk pan-eropa sistem selular radio bergerak yang bekerja pada frekuensi 900 Mhz.

### 2.1.1. Jaringan GSM.

GSM memberikan suatu rekomendasi bukan suatu persyaratan, GSM menspesifikasikan fungsi-fungsi dan antarmuka yang diperlukan secara detail bukan mengarah ke perangkat keras yang digunakan. Alasan tersebut didasari untuk membatasi para *designer* sekecil mungkin namun tetap saja memungkinkan para operator untuk membeli perangkat dari penyedia yang berbeda. Jaringan GSM dibagi menjadi tiga sistem utama: *Switching System* (SS), *Base Station System* (BSS), dan *Operating System and Support* (OSS).

a) *Switching System* (SS).

   Sistem *Switching* bertanggung jawab untuk melakukan proses panggilan dan fungsi pelanggan.

b) *Base Station System* (BSS).

Seluruh fungsi dari radio dilakukan di BSS, dimana terdiri dari *Base Station Controllers* (BSC) dan *Base Transceiver Stations* (BTS).

1. *Base Station Controllers* (BSC).

   BSC menyediakan seluruh fungsi pengawasan dan hubungan fisik antara *Mobile Station Center* (MSC) dan BTS. BSC merupakan *switch* berkapasitas tinggi yang melakukan fungsi sebagai *handover*, data konfigurasi, dan kontrol tingkat daya *Radio Frequency* (RF) di BTS, sejumlah BSC dapat dilayani oleh MSC.

2. *Base Transceiver Station* (BTS).

   BTS menangani antarmuka radio ke *Mobile Station* (MS). BTS adalah perlengkapan radio yang diperlukan untuk melayani setiap panggilan di masing-masing ruang dalam suatu jaringan.

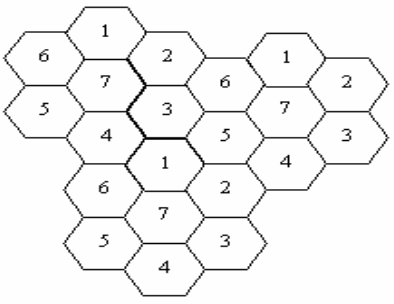
c) *Operating System and Support* (OSS).

*Operating and Maintenance Center* (OMC) tersambung ke seluruh perlengkapan sistem *switching* dan ke BSC, implementasi dari OMC disebut OSS. OSS merupakan wujud fungsional dari pemantauan jaringan operator dan pengontrollan sistem, kegunaan dari OSS adalah untuk menawarkan ke langganan biaya efektif *support* untuk sentralisasi, regional, dan lokal operasional dan aktivitas pemeliharaan dimana diperlukan untuk jaringan GSM, fungsi yang penting dari OSS yaitu memberikan gambaran jaringan dan dukungan aktivitas pemeliharaan dari operasi yang berbeda dan pemeliharaan organisasi.

Bagian paling rendah dari sistem GSM adalah *Mobile Station* (MS). Bagian ini berada pada tingkat pelanggan dan *portable*. Pada tiap sel terdapat *Base Transceiver Station* (BTS). BTS ini berfungsi sebagai *station* penghubung dengan MS dan merupakan sistem yang langsung berhubungan dengan *handphone*. BTS pada dasarnya hanya merupakan "pesuruh" saja. Otak yang mengatur lalu-lintas trafik di BTS adalah *Base Station Controller* (BSC). *Location Updating*, penentuan BTS dan proses *handover* pada percakapan ditentukan oleh BSC ini. Beberapa BTS pada satu region diatur oleh sebuah BSC. Seluruh BSC ini dihubungkan dengan *Mobile Switching Center* (MSC). MSC merupakan pusat penyambungan yang mengatur jalur hubungan antar BSC maupun antara BSC dan jenis layanan telekomunikasi lain (PSTN, operator GSM lain, AMPS, dll).

### 2.1.2. Struktur Sistem Seluler.

Pembagian area dalam kumpulan sel - sel merupakan prinsip penting GSM sebagai sistem telekomunikasi selular. Sel - sel tersebut dimodelkan sebagai bentuk heksagonal seperti pada gambar berikut :




*Gambar 2.1 Pembagian area sel frekuensi GSM*

Tiap sel mengacu pada satu frekuensi pembawa atau kanal tertentu. Pada kenyataannya jumlah kanal yang dialokasikan terbatas, sementara jumlah sel bisa saja berjumlah sangat banyak. Untuk memenuhi hal ini, dilakukan teknik pengulangan frekuensi (*frequency re-use*).

Jelas bahwa semakin besar jumlah himpunan kanal, semakin sedikit jumlah kanal tersedia per-sel dan oleh karenanya kapasitas sistem menurun. Namun, peningkatan jumlah himpunan kanal menyebabkan jarak antara sel yang berdekatan kanal semakin jauh, dan ini mengurangi resiko terjadi interferensi sehingga desain sistem GSM memerlukan kompromi antara kualitas dan kapasitas.

## 2.2. *Short Message Services* (SMS).

SMS merupakan sebuah layanan yang banyak diaplikasikan pada sistem komunikasi tanpa kabel, memungkinkan dilakukannya pengiriman pesan dalam bentuk alphanumerik antara terminal pelanggan atau antara terminal pelanggan dengan sistem eksternal seperti *email, paging* dan lain – lain. SMS pertama kali muncul di belahan Eropa pada sekitar tahun 1991 bersama sebuah teknologi komunikasi *wireless* yang saat ini cukup banyak penggunanya, yaitu *Global System for Mobile Communication* (GSM).

GSM membedakan dua tipe pesan pendek yaitu :

a. Layanan *dedicated*, antara dua partisipan yang membutuhkan sebuah jalur pembawa secara *point to point* ( melalui *Delicated Service Center* ), yang kemudian disebut *point to point* SMS.

*b.* Layanan *broadcast* antara jaringan ( melalui satu atau lebih *Base Station* ) dan semua pengguna dalam sebuah *Service Area*.

### 2.2.1. Elemen dasar jaringan SMS

1. *Short Messaging Entities* (SME), suatu piranti yang dapat menerima dan mengirim pesan pendek.
2. *Short Message Service Center* (SMSC), kombinasi perangkat lunak dan perangkat keras yang brtanggung jawab memperkuat, menyimpan dan meneruskan pesan pendek antar SME dan piranti bergerak (*mobile phone).*
3. *Signaling System 7* (SS7), protokol *signalling* yang umum di gunakan dalam jeringan telepon seluler.
4. *Base Station System* (BSS), berfungsi mengendalikan satu atau lebih BTS dan bertanggung jawab dalam pemberian sumber data dan transmisi sinyal radio elektromagnetis antara MSC dan *mobile phone*.
5. *Home Location Register* (HLR), basis data yang digunakan untuk penyimpanan permanen, pengelolaan dan profil layanan.
6. *Visitor Location Register* (VLR), basis data yang berisi informasi temporal mengenai pelanggan yang berasal dari suatu HLR yang roaming ke HLR lainnya.

### 2.2.2. SMS Center.

Pada saat kita mengirim SMS dari telefon seluler ( *Mobile Originated* ) pesan tersebut tidak langsung dikirimkan ke telefon seluler tujuan ( *Mobile*

*Termiinated* ), akan tetapi dikirim terlebih dahulu ke SMS *Center* (SMSC), baru kemudian pesan tersebut diteruskan ke telefon seluler tujuan. Dengan adanya SMSC ini kita dapat mengetahui status dari SMS yang dikirim, apakah telah sampai atau gagal diterima oleh telefon seluler tujuan. Apabila telefon seluler tujuan dalam keadaan aktif dan dapat menerima SMS yang dikirim, maka akan mengirimkan kembali pesan konfirmasi ke SMSC yang menyatakan bawa pesan telah diterima.

Untuk dapat mengirim dan menerima pesan, harus melakukan koneksi ke SMSC. Ada beberapa cara untuk melakukan koneksi ke SMSC antara lain :

1) Menggunakan terminal baik berupa GSM modem atau telefon seluler. Cara ini adalah yang paling mudah tetapi memiliki kekurangan antara lain jumlah pesan yang dikirim permenit sangat terbatas, untuk mengantisipasi hal ini biasanya digunakan lebih dari satu terminal.
2) Koneksi langsung ke SMSC.

   Dengan melakukan koneksi langsung ke SMSC kita dapat mengirim pesan dalam jumlah banyak. Untuk melakukan koneksi ke SMSC diperlukan protokol penghubung umumnya yang digunakan adalah UCP, SMPP, SIMD2, OIS dan TAP.
3) Menggunakan software bantu.

   Saat ini banyak perusahaan telekomunikasi menawarkan *software* bantu untuk melakukan koneksi ke SMSC, dari yang bersifat *freeware*, *Open Source* sampai dengan komersial.

### 2.2.3. Karakteristik SMS.

Selain sebagai media mengirim dan menerima pesan alphanumerik, SMS juga dapat digunakan sebagai pengangkat muatan biner ( *Binary Payload* ) dan mengimplementasikan tumpukan ( *stack* ) *wap* lewat SMSC. Ada beberapa karakteristik SMS yang penting yaitu :

a. Prinsip kerja dari SMS ini adalah bahwa setiap jaringan mempunyai suatu *Service Center* (SC). Pesan tidak langsung dikirimkan ke tempat tujuan, melainkan disimpan terlebih dahulu di SC menjadi *interface* antara *Public Land Mobile Network* (PLMN).
b. Tranmisi SMS dapat terjadi meskipun *Mobile Station* sedang melakukan komunikasi dengan *Mobile Station* yang lain. Hal ini dimungkinkan karena kanal radio untuk tranmisi *voice* telah ditentukan selama durasi pemanggilan sedangkan SMS merambat pada kanal radio dengan memanfaatkan jalur *signaling*.
c. Pengiriman SMS yang menggunakan kanal *signaling* memiliki dua tipe yaitu :
    1. SMS *point to point* : menyediakan mekanisme untuk mengirimkan pesan hanya dari satu *Mobile Station* (MS) ke MS tertentu, berupa pesan pendek ( *Short Message* ) dan dari piranti bergerak.
    2. SMS *Broadcast* ( *point to multipoint* ) : pengiriman SMS ke beberapa MS sekaligus.
d. Pesan SMS dijamin sampai atau tidak sampai sama sekali, selayaknya *email*, sehingga apabila terjadi kegagalan sistem, *Time Out* (TO) atau hal

lain yang menyebabkan pesan SMS tidak diterima, akan diberikan informasi ( *report* ) yang menyatakan pesan SMS gagal dikirim.

e. Berbeda dengan fungsi *call* ( pemanggilan ), sekalipun saat mengirimkan SMS telefon seluler tidak aktif, bukan berarti pengiriman SMS akan gagal, namum SMS akan masuk ke antrian dulu selama belum TO, SMS akan segera dikirimkan jika telefon seluler sudah aktif.

## 2.3. *Protocol Data Unit* (PDU).

Satuan paket data yang dipertukarkan pada lapisan aplikasi dalam protokol SMS Manager disebut *Protocol Data Unit* (PDU). Dalam protokol tersebut terdapat beberapa macam format PDU dimana penggunaan masing-masing PDU terebut harus sesuai dengan fungsinya, sebagai contoh untuk mengirim sebuah pesan, harus digunakan PDU dengan format *submit_sm, deliver_sm,* atau *data_sm*.

Umumnya cara pengiriman paket data dari satu titik ke titik lain, salah satu titik harus bertindak sebagai server dan titik lainnya sebagai *client*, Inisiatif dan pembentukan sebuah *session* (jenis permintaan atau perintah) dilakukan oleh *client*. Jenis *session* yang dipilih sepenuhnya diserahkan kepada *client* (otorisasi diterima atau tidak tetap dipegang oleh *server*), terdapat tiga buah *session* yang dapat dipilih yaitu:

1. *Receiver* (RX), bila *client* ingin dapat menerima paket data.
2. *Transmitter* (TX), bila *client* ingin dapat mengirimkan paket data.
3. *Transceiver* (TRX), bila *client* ingin dapat mengirim dan menerima paket data.

### 2.3.1. Format Umum *Protocol Data Unit (PDU)*.

Secara umum format PDU dalam protokol SMS Manager adalah sebagai berikut:

**Tabel 2.1. Format PDU**

| | SMPP PDU Field | Size (Octets) | Type | Description |
|---|---|---|---|---|
| **PDU *HEADER*** | *Command_length* | 4 | Integer | *Overall size of* PDU *Including header and body* |
| | *Command_id* | 4 | Integer | *Identifies the* PDU |
| | *Command_status* | 4 | Integer | *Use to carry an error code* |
| | *Sequence_number* | 4 | Integer | *Use to uniquely identified* PDU *session* |
| ***BODY*** | *Standard Parameters* | Var. | Mixed | *The body part of* PDU |
| | TLV Parameter | Var. | mixed | |

Enambelas oktet pertama disebut PDU *header*, merupakan bagian yang bersifat mandatori atau harus selalu ada dalam setiap PDU. Sisanya disebut PDU *Body,* merupakan bagian yang bersifat opsional (tergantung jenis PDU). *Body* PDU terdiri dari parameter standar dan parameter tambahan yang disebut parameter *Tag-Length_Value* (TLV), berikut penjelasan masing-masing bagian tersebut :

1. *Command_Length*.

   *Command_length* menyatakan panjang yang sesungguhnya dari PDU meliputi PDU *Header* dan PDU *Body*, termasuk didalamnya oktet dari *command_length* sendiri. Posisinya terletak pada awal atau field pertama dari setiap PDU.

2. *Command_id*.

*Command_id* merupakan parameter yang mendgindikasikan jenis operasi dari protokol SMS Manager yang sedang dilakukan. Parameter ini dikodekan dalam 4 oktet dengan tipe data integer.Nilai parameter ini berkisar antara 0x00000000 hingga 0x00000001FF untuk parameter yang termasuk kategori *request*, kategori respons nilainya berkisar antara 0x80000000 hingga 0x800001FF.

3. *Command_status*.

   *command_satus* merupakan paramater yang menunjukkan status operasi dalam protokol dan hanya relevan untuk PDU-PDU kategori *respons*. Untuk PDU-PDU kategori *request*, parameter ini diabaikan.

4. *Sequence_number*.

   *sequence_number* merupakan parameter yang digunakan untuk menandai sebuah PDU secara unik agar dapat dibedakan dengan PDU lain.

5. *Parameter Standar*.

   *Parameter standar* merupakan parameter yang terdapat pada PDU *Body* pada PDU-PDU tertentu (PDU yang *non-header only*, yaitu PDU *submission* dan *delivery*). Formatnya sendiri bervariasi tergantung PDU-nya dan merupakan kombinasi data dengan tipe integer.

6. Parameter TLV.

   *Tag-Length-Value*, merupakan parameter dalam protokol yang dapat dibongkar-pasang sesuai dengan kebutuhan. Satu-satunya syarat hanyalah parameter ini, bila disertakan, harus diletakkan setelah paramater standar.

Berikut merupakan contoh Strutuktur yang dikirim PDU

07 91 2658050000F0 11 00 0C 91 265836164900 00 00 FF 05 E8329BFD06

**Tabel 2.2. Struktur yang di kirim PDU**

| Oktet / Digit Hexa | Keterangan |
|---|---|
| 07 | Panjang atau jumlah pasangan digit dari nomor SMSC (service number) yang digunakan, dalam hal ini adalah 7 pasangan (14 digit berikutnya) |
| 91 | Jenis nomor SMSC. Angka 91 menandakan format nomor internasional (misal +6281xxx). Untuk 081xxx menggunakan angka 81 |
| 2658050000F0 | Nomor SMSC yang digunakan. Karena jumlah digit nomor SMS adalah ganjil, maka digit paling belakang dipasangkan dengan huruf F. Kalau diterjemahkan, nomor SMSC yang digunakan adalah +62855000000 (IM3) |
| 11 | Oktet pertama untuk PDU SMS untuk dikirim |
| 00 | TP-*Message-Reference*. Diisi "00" agar diisi otomatis oleh handphone. |
| 0C | Panjang digit dari nomor penerima (0C hex = 12 desimal) |
| 91 | Jenis nomor penerima (sama dengan jenis nomor SMSC) |
| 265836164900 | Nomor penerima SMS, yang jika diterjemahkan adalah +628563619400 |
| 00 | Pengenal protokol, dalam hal ini adalah 0 |
| 00 | Skema pengkodean SMS, juga bernilai 0. |
| FF | Validitas waktu. FF berarti maksimum |
| 05 | Panjang dari pesan SMS, dalam hal ini adalah 5 huruf (dalam mode 7 bit) |
| E8329BFD06 | Pesan SMS dalam mode 7 bit. Jika diterjemahkan kedalam 8 bit, lalu dirubah ke ASCII, maka didapat pesan 'hello' |

### 2.3.2. Contoh Penggunaan PDU

1) Langkah-langkah untuk menentukan kode PDU yang dibutuhkan untuk mengirimkan *(send)* SMS ke SMS Center :

a. 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 tak memiliki pasangan maka angka tersebut dipasangkan dengan huruf F didepannya.

Contoh konversi nomor ponsel kedalam kode PDU dengan menggunakan :

1. Kode Nasional.

   SMS *Center* menggunakan nomor 0856000000 maka :

   1.a. Terdapat 6 pasang angka heksa desimal.

   1.b. Kode nasional 81, ada 1 pasang

   1.c. 80-56-00-00-00

   Maka kode PDU yang didapat : 06818056000000

2. Kode Internasional.

   SMS *Center* menggunakan nomor 62811000000, maka:

   2.a. Terdapat 7 pasang angka heksa desimal

   2.b. Kode internasional 91, ada 1 pasang

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

   Maka kode PDU yang didapat : 07912618010000F0

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 : 081321272326, maka konversi kedalam heksa desimal yaitu dengan cara:

   1. Kode Nasional.

      1.a. terdapat 12 angka, berarti dalam heksa adalah 0C

      1.b. kode nasional adalah 81

      1.c 80-31-12-72-32-62

      Maka kode *subheader* PDU yang didapat adalah : 0C81803112723262

   2. Kode Internasional.

      2.a 62 81 32 12 17 32 6

      2.b. Terdapat 13 angka, berarti dalam heksa adalah 0D

      2.c. Kode internasional adalah 91

      2.d. 261823212732F6

      Maka kode *subheader* PDU yang didapat adalah : 0D91261823212732F6

e. Bentuk SMS.

   00 : dikirim sebagai SMS

01 : dikirim sebagai Telekomunikasi

f. Skema *encoding* data *Input* dan *Output* (I/O).

Terdapat dua skema yaitu ;

1. Skema 7 bit, ditandai dengan angka 00
2. Skema 8 bit, ditandai lebih besar dari nol (dalam heksa)

g. Jangka Waktu atau validasi sebelum SMS *expired* atau kadaluarsa, jika bagian ini diloncat 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.

**Tabel 2.3. *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 |

h. Isi SMS.

Terdiri dari dua *subheader* yaitu:

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

**Tabel 2.4 Konversi heksa desimal**

| | | | | | **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 |
| **Hex** | **B4** | **B3** | **B2** | **B1** | | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| 0 | 0 | 0 | 0 | 0 | **0** | @ | Δ | SP | 0 | - | P | | p |
| 1 | 0 | 0 | 0 | 1 | **1** | | | ! | 1 | A | Q | a | q |
| 2 | 0 | 0 | 1 | 0 | **2** | $ | Φ | “ | 2 | B | R | b | r |
| 3 | 0 | 0 | 1 | 1 | **3** | | Γ | # | 3 | C | S | c | s |
| 4 | 0 | 1 | 0 | 0 | **4** | | | | 4 | D | T | d | t |
| 5 | 0 | 1 | 0 | 1 | **5** | | | % | 5 | E | U | e | u |
| 6 | 0 | 1 | 1 | 0 | **6** | | | & | 6 | F | V | f | v |
| 7 | 0 | 1 | 1 | 1 | **7** | | | . | 7 | G | W | g | w |
| 8 | 1 | 0 | 0 | 0 | **8** | | | ( | 8 | H | X | h | x |
| 9 | 1 | 0 | 0 | 1 | **9** | | | ) | 9 | I | Y | i | y |
| A | 1 | 0 | 1 | 0 | **10** | LF | | * | : | J | Z | j | z |
| B | 1 | 0 | 1 | 1 | **11** | | | + | ; | K | | k | |
| C | 1 | 1 | 0 | 0 | **12** | | | ‘ | < | L | | l | |
| D | 1 | 1 | 0 | 1 | **13** | CR | | _ | = | M | | m | |
| E | 1 | 1 | 1 | 0 | **14** | | | | > | N | | n | |
| F | 1 | 1 | 1 | 1 | **15** | | | / | ? | O | | o | |

Contoh konversi *Text* ke bentuk PDU dan PDU ke bentuk *Text*

Kata “hello”

*Text* ke bentuk PDU :

**Tabel 2.5 Konversi *Text* ke PDU**

| h | e | l | l | o |
|---|---|---|---|---|
| 1101000 | 1100101 | 1101100 | 1101100 | 1101111 |
| 11101000 | 00110010 | 10011011 | 11111101 | 00000110 |
| E8 | 32 | 9B | FD | 06 |

Maka konversi dari kata “hello”kedalam bentuk PDU adalah: E8329BFD06

PDU ke bentuk *Text* :

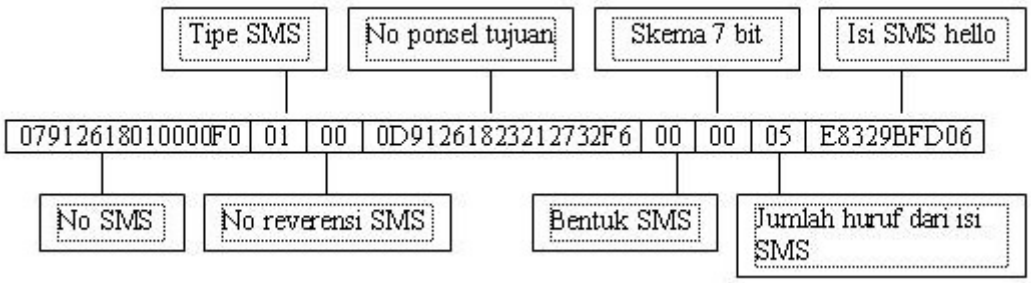
**Tabel 2.6 Konversi PDU ke *Text***

| E8 | 32 | 9B | FD | 06 |
|---|---|---|---|---|
| 11101000 | 00110010 | 10011011 | 11111101 | 00000110 |
| 1101000 | 1100101 | 1101100 | 1101100 | 1101111 |
| h | e | l | L | o |

Maka konversi dari PDU "E8329BFD06" kedalam bentuk *text* adalah: hello

Setelah selesai dari langkah a sampai dengan langkah h, maka tinggal menggabungkan *subheader* - *subheader* tersebut, sehingga didapat PDU untuk mengirim SMS :

07912618010000F0 01 00 0D91261823212732F6 00 00 05 E8329BFD06




*Gambar 2.2 Contoh SubHeader kode PDU*

Kode PDU ini akan diterima oleh SMS *Center* dan kemudian mengirimkannya ke nomor tujuan

2) Langkah-langkah untuk menentukan *subheader* PDU untuk menerima (*receive)* SMS tidak begitu jauh berbeda dari pengiriman SMS yaitu terdiri dari:

a. Nomor SMS *Center*.

b. Tipe SMS, tipe SMS untuk menerima adalah 04.

c. Nomor ponsel pengirim.

d. Bentuk SMS.

e. Skema Encoding.

f. Validitas tanggal dan waktu SMS, jika tak dibatasi dilambangkan dengan 00.

g. Batas waktu validasi.

h. Isi SMS.

## 2.4. Perangkat Lunak.

### 2.4.1. Pengertian Perangkat Lunak.

Perangkat lunak terdiri dari program, dokumen dan data yang berisi perintah atau program komputer yang bila dieksekusi akan memberikan fungsi dan unjuk kerja seperti yang diinginkan.

Perancangan perangkat lunak adalah disiplin manajerial dan teknis yang berkaitan dengan pembuatan dan pemeliharaan produk perangkat lunak secara sistematis, termasuk pengembangan dan modifikasinya, yang dilakukan pada waktu yang tepat dan dengan mempertimbangkan faktor biaya.

*Software engineer* bertugas melakukan analisa, rancangan, uji dan verifikasi, dokumentasi, pemeliharaan perangkat lunak, serta pengelolaan proyek. *Software engineer* harus mempunyai keterampilan dan pengalaman seorang *programmer*. *Programmer* adalah individu yang bertugas dalam hal rincian implementasi, pengemasan, dan modifikasi algoritma serta struktur data, dituliskan dalam sebuah bahasa pemrograman tertentu.

### 2.4.2. *Life Cycle* Perangkat Lunak.

Siklus hidup (*Life-cycle)* sebuah perangkat lunak mencakup semua kegiatan yang perlu dilakukan untuk mendefinisikan, mengembangkan, menguji, mengantarkan, mengoperasikan, dan memelihara produk perangkat lunak.

Beberapa model tersebut yaitu :

1. *Phased Model* (*Model Fase*).

*Model* ini membagi *life cycle* ke dalam sederetan kegiatan (*fase*). Setiap *fase* membutuhkan informasi masukan, proses, dan produk yang terdefinisi dengan baik. Deretan *fase* tersebut adalah :

a. Analisa, terdiri dari *fase* pertama yaitu perencanaan yang menghasilkan dua produk yaitu Pendifinisian Sistem (*System Definition)* dan Perencanaan Proyek (*Poject Plan)* dan *fase* kedua yaitu penetapan persyaratan yang menghasilkan sebuah produk spesifikasi kebutuhan perangkat lunak (*Software Requirements Specifications).*
b. Perancangan, melakukan identifikasi terhadap komponen perangkat lunak (fungsi, arus data, penyimpanan data), hubungan antar komponen, struktur perangkat lunak (dekomposisi menjadi modul-modul dan antar muka Perangkat Lunak). Fase ini menghasilkan arsitektur rinci, terutama dalam bentuk algoritma-algoritma.
c. Implementasi, adalah terjemahan langsung arsitektur rinci ke dalam bahasa pemrograman tertentu.
d. Pengujian, terdiri dari *fase* pertama yaitu uji integrasi dengan melakukan pengujian terhadap semua modul dan pengantarmukaan sehingga pada

*level* sistem dapat beroperasi dengan benar, dan fase kedua yaitu uji penerimaan dengan melakukan berbagi pengujian mengacu kepada berbagai persyaratan yang telah ditentukan.

e. Pemeliharaan, terdiri dari *fase* peningkatan kemampuan, adaptasi terhadap lingkungan pemrosesan, dan melakukan berbagai koreksi atas kesalahan yang terjadi.

### 2.4.3. Kualitas Perangkat Lunak.

Sebuah perangkat lunak yang berkualitas diukur dari beberapa atribut yaitu :

a. Kegunaan, yaitu pemenuhan terhadap kebutuhan pengguna.
b. Keandalan, yaitu kemampuan melaksanakan fungsi yang diinginkan.
c. Kejelasan, yaitu penulisan program dilakukan secara jelas dan mudah dimengerti.
d. Efisiensi, terutama dalam waktu eksekusi dan penggunaan *memory.*

Menurut Robert Glass suatu *software* memenuhi syarat berkualitas jika *software* yang dihasilkan :

1. Sesuai dengan kebutuhan.
2. *User Friendly.*
3. *Up to Date.*
4. Manajemen file dan database terorganisir dengan baik.
5. Tepat waktu dan sesuai anggaran.
6. Adanya kepuasan dari pelanggan**.**

## 2.5. Basis Data (*Database*)

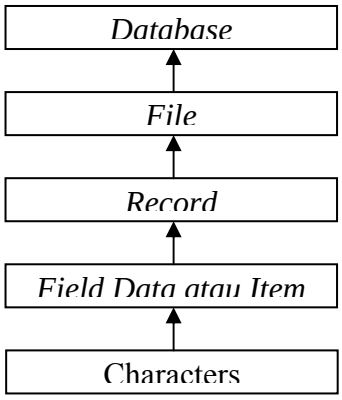
Database merupakan kumpulan dari *item* data yang saling berhubungan satu dengan yang lainnya yang diorganisasikan berdasarkan sebuah skema atau struktur tertentu, tersimpan di *hardware* komputer dan dengan *software* untuk melakukan manipulasi untuk kegunaan tertentu.

Beberapa alasan mengapa diperlukan *database*, yaitu :

1) Salah satu komponen penting dalam *sistem informasi*, karena merupakan dasar dalam menyediakan informasi
2) Menentukan kualitas informasi : akurat, tepat pada waktunya dan relevan. Informasi dapat dikatakan bernilai bila manfaatnya lebih efektif dibandingkan dengan biaya mendapatkannya
3) Mengurangi duplikasi data (data r*edudanc*y)
4) Hubungan data dapat ditingkatkan (data *relatability*)
5) Mengurangi pemborosan tempat simpanan luar

### 2.5.1. Tingkatan *Database*

Berikut merupakan tingkatan dari dari *Database*




*Gambar 2.3 Tingkatan Database*

Keterangan :

a. *Characters* : merupakan bagian data yang terkecil, dapat berupa karakter numerik, huruf ataupun karakter-karakter khusus (*special characters*) yang membentuk suatu *item* data atau *field*.

b. *Field* : merepresentasikan suatu *atribut* dari *record* yang menunjukkan suatu *item* dari data, seperti misalnya nama, alamat dan lain sebagainya. Kumpulan dari *field* membentuk suatu *record*.

   a} *Field Name*: harus diberi nama untuk membedakan *field* yang satu dengan lainnya

   b} *Field* representation: *tipe field* (karakter, teks, tanggal, angka, dsb), lebar *field* (ruang maksimum yang dapat diisi dengan karakter-karakter data)

   c) *Field value*: isi dari *field* untuk masing-masing *record*.

c. *Record* : Kumpulan dari *field* membentuk suatu *record*. *Record* menggambarkan suatu *unit* data individu yang tertentu. Kumpulan dari *record* membentuk suatu *file*. Misalnya *file* personalia, tiap-tiap *record* dapat mewakili data tiap-tiap karyawan.

d. *File* : *File* terdiri dari *record-record* yang menggambarkan satu kesatuan data yang sejenis. Misalnya *file* mata pelajaran berisi data tentang semua mata pelajaran yang ada.

e. *Database* : Kumpulan dari *file* atau *tabel* membentuk suatu database

### 2.5.2. *Database Management System* (DBMS)

DBMS adalah kumpulan *database* (basis data) beserta aplikasi yang digunakan untuk mengolah basis data tersebut. DBMS biasanya menawarkan kemampuan pengolahan data secara terintegrasi. Kemampuan DBMS antara lain :

1. Membuat, menghapus, menambah, dan memodifikasi data
2. Biasanya dilengkapi dengan *fitur* berbasis *windows* yang akan memudahkan pengguna dalam pengolahan data
3. Memberikan keamanan bagi data sehingga tidak semua orang dapat mengakses data yang ada (*login* dan *password*)
4. Kemampuan berkomunikasi dengan program aplikasi lain. Misalnya dengan *visual basic* kemampuan mengakses melalui komunikasi antar komputer (*client server*)

## 2.6. 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 batere, kekuatan sinyal, nama operator, lokasi dan *cell* ID.

**Tabel 2.7. Perintah AT-*Command* GSM 07.07**

| AT-*Command* | Singkatan | Fungsi |
|---|---|---|
| AT+C..... | *Attention + Command*..... | Awal perintah hayes untuk.... |
| AT+CGMI | *Get Manufactured* Identification | Menampilkan ID pabrik pembuat |
| AT-CGMM | *Get Mobile Model* | Menampilkan ID model *Handphone* |
| AT+CGSN | *Get Serial Number* | Menampilkan serial nomor IMEI |
| AT+COPS | *Operator Selection* | Memilih operator jaringan |
| AT+CLCK | *Lock* | Merubah penguncian telepon |
| AT+CPWD | *Change Password* | Merubah *password* |
| AT+CLIP | *List Incoming Phone* | Menampilkan nomor telepon masuk |
| AT+CPIN | *Pin* | Memasukan PIN |
| AT+CBC | *Battery Charge* | Mengisi batere |
| AT+CSQ | *Signal Quality* | Menampilkan kualitas sinyal |
| AT+CPBW | *Phone Book Write* | Memasukan data pada buku telepon |
| AT+CRSM | *Restricted SIM Access Mode* | Memblokir akses SIM |
| AT+CAOC | *Advice Of Charge* | Menampilkan biaya panggilan |
| AT+CCLK | *Clock Mode* | Pengaturan Jam |
| AT+CALM | *Alert Sound Mode* | Pengaturan jenis nada dering |
| AT+CRSL | *Ringer Sound Level* | Pengaturan volume nada dering |
| AT+CMUT | *Mute Control* | Pengaturan tanpa suara |
| AT+CLVL | *Level Volume* | Pengaturan volume pengeras suara |
| AT+CVIB | *Vibration Mode* | Pengaturan nada getar |
| AT+CPBR | *Phone Book Read* | Menampilkan daftar buku telepon |
| AT+CLCC | *List Curent Call* | Menampilkan penelepon terakhir |

## 2.7. Metodologi yang digunakan

Metodologi yang digunakan adalah *Object Oriented*. dengan menggunakan *Unified Modeling Language (UML)*, Sehingga diharapkan hasil yang akan dikeluarkan tidak jauh dari apa yang telah dirancang.

### 2.7.1. Konsep Dasar Orientasi Objek (7)

Romi Satria Wahono menjelaskan pengertian *Object Orientation* sebagai berikut :

a. Teknologi baru yang didasarkan pada objek-objek dan kelas–kelas.
b. Cara berfikir untuk mengorganisir *software* sebagai kumpulan objek yang bekerjasama dengan struktur data dan kelakuan *(behaviour)*.
c. Sebuah abstraksi dari dunia nyata berdasarkan objek dan keterkaitan mereka dengan objek lain.

Pengertian Objek itu sendiri menurut Romi harus mencakup :

a. Objek adalah benda atau konsep. Bisa benda dunia nyata atau konsep atau abstraksi dari benda atau konsep dijelaskan sebagai representasi dari *software*.
b. Objek memiliki atribut dan kelakuan (*behavior*).

Sedangkan Kelas sendiri adalah :

a. Kelas adalah deskripsi dari koleksi objek objek dengan kesamaan atribut dan kelakuan.
b. Secara singkatnya, pengertian atau spesifikasi dari kelas mencakup pengertian dari atribut *comprising the state, methode*

mengimplementasikan kelakuan (*behavior*), dan bagaimana menangani pembuatan dan penghancuran dari sebuah objek.

### 2.7.2. Konsep Dasar *Unified Modeling Language* (UML)

Sejarah UML sendiri cukup panjang dimulai pada bulan Oktober 1994 Booch, Rumbaugh dan Jacobson, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 di*release draft* pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh *Object Management Group*.

Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. Booch, Rumbaugh dan Jacobson menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek.

Sedangkan Sri Dharwiyanti dan Romi Satria Wahono menjelaskan *Unified Modelling Language* (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Diagram–diagram yang digunakan untuk medefinisikan UML adalah sebagai berikut :

***a. Use Case Diagram***

*Use case diagram* menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana".

Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya *login* ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya.

Seorang atau sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal.

Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*.

Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behavior*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

***b. Class Dagram***

*Class* adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut atau properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda atau 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. Nama (dan *stereotype*).
2. Atribut.
3. Metoda.

Atribut dan metoda dapat memiliki salah satu sifat berikut :

1. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
2. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
3. *Public*, dapat dipanggil oleh *class* mana saja.

*Class* dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung

diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*.

Sesuai dengan perkembangan *class model*, *class* dapat dikelompokkan menjadi *package*. Kita juga dapat membuat diagram yang terdiri atas *package*. Adapun hubungan antar *Class* yang dapat terbentuk antara lain:

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian ("terdiri atas..").
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.

### *c. Statechart Diagram*

*Statechart diagram* menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima.

Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).

Dalam UML, *state* digambarkan berbentuk segi empat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring. Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.

***d. Activity Diagram***

*Activity diagrams* menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

*Activity diagram* merupakan *state diagram* khusus, dimana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan *behaviour internal* sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari *level* atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity diagram* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

***e. Sequence Diagram***

*Sequence diagram* menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

*Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada *fase* desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*. Untuk objek-objek yang memiliki sifat

khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary, controller* dan *persistent entity*.

### *f. Collaboration Diagram*

*Collaboration diagram* juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*. Setiap *message* memiliki *sequence number*, dimana *message* dari level tertinggi memiliki nomor 1. *Messages* dari *level* yang sama memiliki *prefiks* yang sama.

### *g. Component Diagram*

*Component diagram* menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya. Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time, link time*, maupun *run time*.

Umumnya komponen terbentuk dari beberapa *class* dan atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

### *h. Deployment Diagram*

*Deployment/physical diagram* menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, *server* atau piranti keras), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi *server*, dan hal-hal lain yang bersifat fisikal.

Sebuah *node* adalah *server*, *workstation*, atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.