BAB 2

LANDASAN TEORI

2.1 Teori-teori Umum

2.1.1. Teori-teori *Good Organization Governance* (GOG)

Definisi *Governance* menurut *United Nation Economic and Social Commision for Asia Pacific* (UNESCAP) adalah proses pengambilan keputusan dan proses dengan mana keputusan diimplementasikan (atau tidak diimplementasikan). Sedangkan *Good Organization Governance* bila diterjemahkan kedalam bahasa Indonesia memiliki arti "tata kelola organisasi yang baik", yang merupakan hasil dari terapan *Good Governance* kedalam organisasi.

Good Governance menurut *United Nation Economic and Social Commision for Asia Pacific* (UNESCAP) memiliki 8 karakteristik utama, yaitu :

1. *Participation*

   Partisipasi dari seluruh anggota baik pria maupun wanita merupakan landasan utama dari *Good Governance*. Partisipasi dapat berupa partisipasi langsung atau melalui institusi yang sah atau melalui perwakilan. Hal ini menunjukkan bahwa kekhawatiran yang paling rentan dalam masyarakat akan dipertimbangkan dalam pengambilan suatu keputusan. Partisipasi perlu diinformasikan dan terorganisir

yang berarti kebebasan berserikat dan berekspresi di satu sisi dan masyarakat sipil yang terorganisir di sisi lain.

2. *Rule of Law*

   Good Governance memerlukan kerangka hukum yang legal dimana kerangka ini tidak memprioritaskan pihak manapun. Hal ini juga memerlukan perlindungan penuh terhadap hak asasi manusia, terutama kaum minoritas. Penegakan hukum yang netral membutuhkan pengadilan yang independen dan kepolisian yang tidak memihak.

3. *Transparency*

   Keputusan yang diambil dan penegakannya dilakukan dengan cara mengikuti aturan dan regulasi yang ada. Hal ini juga berarti informasi tersedia bebas dan bisa diakses oleh mereka yang dipengaruhi oleh keputusan ini. Informasi ini harus disediakan dalam bentuk yang mudah dimengerti.

4. *Responsiveness*

   Good Governance mensyaratkan bahwa segala institusi dan proses yang ada harus melayani semua *Stakeholder* dengan jangka waktu yang wajar.

5. *Consensus Oriented*

   Ada beberapa aktor dan sebagai titik pandang dalam suatu masyarakat. *Good Governance* membutuhkan mediasi dari kepentingan-kepentingan yang ada dalam masyarakat untuk mencapai konsensus yang luas berdasarkan kepentingan yang paling

mendesak dari seluruh masyarakat dan bagaimana hal ini dapat dicapai. Hal ini juga memerlukan perspektif yang luas dengan jangka waktu yang panjang atas apa yang dibutuhkan oleh pengembangan manusia yang berkelanjutan dan bagaimana cara untuk mencapai pembangunan-pembangunan tersebut. Hal ini merupakan hasil dari pemahaman atas sejarah, budaya, dan konteks sosial dari suatu masyarakat.

6. *Equity and inclusiveness*

   Kesejahteraan masyarakat tergantung dari bagaimana para anggotanya merasa berperan dalam masyarakat tersebut.

7. *Effectiveness and Efficiency*

   Good Governanace berarti semua proses dan institusi memproduksi hasil yang memenuhi kebutuhan masyarakat dengan menggunakan sumber daya mereka sebaik-baiknya. Konsep efisiensi *Good Governance* juga mencangkup peggunaan sumber daya alam yang berkelanjutan dan perlindungan lingkungan.

8. *Accountability*

   Akuntabilitas merupakan kebutuhan utama dari *Good Governance* baik dilembaga pemerintahan, swasta ataupun organisasi masyarakat. Segala sesuatu harus bisa dipertanggungjawabkan kepada publik dan kepada para *Stakeholder*. Siapa yang bertanggungjawab, tergantung kepada keputusan maupun tindakan yang diambil. Biasanya suatu organisasi atau institusi bertanggungjawab kepada mereka yang

dipengaruhi oleh keputusan tersebut. Akuntabilitaas tidak dapat ditegakkan tanpa transparasi dan aturan hukum.

2.1.2. Teori-teori Sistem Basis Data

2.1.2.1. Basis Data

Menurut Connolly dan Begg (2002, p14), basis data adalah suatu koleksi bersama data-data yang saling terkait secara logis, dan juga merupakan pendeskripsian dari data-data tersebut, yang dirancang untuk menyajikan informasi yang dibutuhkan oleh sebuah organisasi. Menurut Whitten et al.(2004, p548), basis data adalah kumpulan *file* yang saling terkait. Berdasarkan pendapat para ahli diatas, maka basis data adalah kumpulan data yang saling berhubungan secara logikal yang dapat digunakan untuk membantu dalam pengambilan keputusan pada sebuah organisasi atau perusahaan.

Dalam basis data, terdapat tiga istilah penting, yakni entitas, atribut, dan *relationship*. Entitas adalah sebuah objek berbeda (bisa seseorang, tempat, sesuatu, konsep, ataupun kejadian) dalam organisasi yang harus direpresentasikan dalam basis data. atribut adalah sebuah property yang mendeskripsikan beberapa aspek dari objek yang ingin di-*record*. *Relationship* adalah sebuah asosiasi antar entitas (Connolly dan Begg, 2002, p15).

2.1.2.2. *Database Management System* (DBMS)

Menurut Connolly dan Begg (2002, p16), *Database Management System* (DBMS) merupakan sistem *software* untuk mendefinisikan, membuat, dan memelihara *database* dan menyediakan akses terkontrol untuk *database* yang berkaitan. DBMS juga merupakan *software* yang berinteraksi dengan aplikasi program dan *database* itu sendiri. DBMS menyediakan beberapa fasilitas seperti mengijinkan pengguna untuk mendefinisikan *database* yang biasa disebut dengan *Data Definition Language (DDL)*. DDL memungkinkan pengguna untuk menspesifikasikan tipe data, struktur dan batasan-batasan data. semua spesifikasi tersebut akan disimpan di dalam *database*, memperbolehkan pengguna untuk memasukkan data (*insert*), mengubah data (*update*), menghapus data (*delete*), dan mengambil data (*retrieve*) dari *database* yang dikenal dengan istilah *Data Manipulation Language (DML)*. DML merupakan fasilitas pengadaan umum mengenai data dengan menggunakan *query language*. Selain itu, fasilitas DBMS lainnya adalah menyediakan kontrol dalam mengakses *database* meliputi sistem keamanan yang dimana menolak pengguna yang tidak berwenang dalam mengakses data, sistem integritas yang dimana mengatur konsistensi penyimpanan data, sistem dari data kontrol konkurensi yang dimana membolehkan mengakses data secara bersama-sama, sistem kontrol *recovery* adalah dengan mengembalikan *database*

yang disebabkan oleh kesalahan pada *hardware* dan *software*, dan katalog yang dapat diakses oleh pengguna yang dimana memuat deskripsi data didalam *database*.

Menurut Connolly dan Begg (2002, p18-23), komponen dari DBMS adalah :

1. *Hardware*

   DBMS dan applikasi *database* membutuhkan *hardware* untuk dapat dieksekusi. *Hardware* dapat berupa personal komputer (PC) sampai dengan *mainframe* atau jaringan komputer. Sebagian *hardware* bergantung dengan permintaan atau kebutuhan organisasi dan DBMS yang digunakan. DBMS membutuhkan kapasitas yang besar dalam menyimpan data.

2. *Software*

   Komponen dari *software* mendukung perfoma *software* DBMS, sistem operasi dan *network software* serta program aplikasi lainnya.

3. Data

   Data merupakan salah satu komponen DBMS yang paling penting. *Database* memuat data operasional dan metadata. Struktur dari *database* disebut dengan skema. Data pada sebuah sistem *database* baik sistem satu pengguna maupun sistem banyak pengguna harus terintegrasi dan dapat digunakan secara bersama-sama. Kebanyakan digunakan oleh organisasi dan deskripsi data.

4. Prosedur

   Prosedur merupakan isntruksi dan aturan yang harus disertakan dalam merancang dan menggunakan *database*. Beberapa instruksi-instruksi dalam merancang *database* dan DBMS adalah:

   a. *Log on* ke dalam DBMS.
   b. Menggunakan sebagian fasilitas dari DBMS atau aplikasi program.
   c. Memulai dan menghentikan DBMS.
   d. Membuat *backup database*.
   e. Mengstasi atau dapat mengendalikan kesalahan *hardware* atau *software*. Dalam hal ini instruksi meliputi mengidentifikasikan kesalahan komponen, mencari solusi untuk memperbaiki kesalahan komponen dan *recovery database*.
   f. Mengubah struktur tabel, mengorganisasikan *database* seperti memperbaiki performa dan meletakkan data ke tempat penyimpanan lainnya.

5. Pengguna

   Database dan DBMS memerlukan sumber daya manusia untuk mengatur jalannya mekanisme *database* dan DBMS. Macam-macam yang dapat dikategorikan untuk mengatur jalannya proses *database* dan DBMS adalah :

a) DA (*Data Administrator)*, seseorang yang berwenang untuk membuat keputusan strategis dan kebijakan mengenai data yang ada.

b) DBA (*Database Administrator*), menyediakan dukungan teknis untuk mengimplentasikan keputusan strategis tersebut dan bertanggun jawab atas seluruh sistem kontrol pada level teknis.

c) *Database designers* (*Logical and Physical*), lebih menekankan pada identifikasi data, hubungan antara data, batasan-batasan data sebelum dimasukkan kedalam *database*.

d) *Application Developers*, bertanggung jawab untuk membuat aplikasi *database* dengan menggunakan bahasa pemrograman yang ada seperti C++, Java, dan sebagainya.

e) *End Users*, adalah siapapun yang berinteraksi dengan sistem secara online melalui *workstation* atau terminal.

Adapun keuntungan penggunaan DBMS antara lain (Connolly dan Begg, 2002, p25):

- Kontrol data redudansi / mengurangi data rangkap
- Konsistensi data
- Tambahan informasi dari data yang sama
- Meningkatkan integritas data
- Meningkatkan sistem keamanan

- Meningkatkan produktivitas
- Peningkatan layanan *recovery* dan *back up* semakin baik

Selain itu, terdapat beberapa kerugian atau kelemahan dari DBMS antara lain (Connolly dan Begg, 2002, p29):

- Rumit atau kompleks
- Ukuran
- Biaya DBMS
- Biaya tambahan *hardware*
- Biaya konversi
- Performa
- Dampak yang lebih besar pada kegagalan

2.1.2.3. *Entity Relationship Diagram* (ERD)

Menurut Connolly dan Begg, (2002, p355), *Entity Type* adalah sekumpulan objek yang memiliki properti-properti yang sama, dimana sekumpulan objek ini diidentifikasi oleh perusahaan sebagai hal yang memiliki ekstensi yang independen, sedangkan *Relationship Type* adalah sekumpulan asosiasi *entity type* yang memiliki arti. Dari pengertian di atas dapat disimpulkan bahwa *Entity Relationship Diagram* (ERD) merupakan diagram yang menggambarkan asosiasi yang memiliki arti dari sekumpulan objek yang memiliki properti-properti yang sama.

2.1.3. Internet

Menurut Douglas E. Comer (*Computer Networks and Internets with Internet Applications*, p657), internet adalah sekumpulan jaringan yang tersambung dengan banyak router yang dikonfigurasikan untuk melewati trafik diantara setiap komputer yang terhubung ke sekumpulan jaringan tersebut. Internet telah bertumbuh yang berawal dari prototipe penelitian menjadi sebuah sistem komunikasi global yang menjangkau semua bangsa di dunia. Saat ini internet sudah banyak digunakan baik sebagai media penyedia informasi, melakukan bisnis bahkan sebagai media untuk melakukan sosialisasi dengan orang lain.

Internet berawal di tahun 1969 dengan nama ARPANET (ARPA merupakan singkatan dari *Advanced Research Project Agency* dari Departement Pertahanan AS) yang berupa sambungan empat komputer pada kontraktor pertahanan dan universitas yang berbeda. Dari sini, jaringan meluas menjadi 62 komputer pada tahun 1974, 500 komputer pada tahun 1983 dan berjumlah 28.000 pada tahun 1987. Akan tetapi, jaringan komputer saat itu masih didominasi untuk kepentingan riset dan akademik. Segala sesuatu masih bebasis teks-tak ada gambar, video, maupun suara. Dan semua berubah pada awal tahun 1990 ketika dimulainya era *World Wide Web*, yaitu ketika internet bisa berisi multimedia. Saat itu untuk pertama kalinya, *browser* (untuk mencari halaman web) diperkenalkan.

2.1.4. www

www merupakan singkatan dari *World Wide Web* yaitu sistem *hypermedia* yang digunakan pada internet dimana halaman informasi dapat mengandung text, gambar, audio, video, dan *link* ke halaman lain (Douglas E. Comer, 2004, p681). Internet memang telah hadir lebih dari 40 tahun yang lalu, tetapi satu hal penting yang mempopulerkan internet, selain email, adalah World Wide Web atau "web" yang mulai dikembangkan awal tahun 1990-an. Web didefinisikan sebagai sistem interkoneksi komputer internet (disebut server) yang mendukung dokumen-dokumen berformat multimedia. Kata multimedia yang berarti "banyak media" berkaitan dengan teknologi yang menyajikan informasi di lebih dari satu media, misalnya teks, gambar, suara, video. Dengan kata lain, web menyediakan informasi dalam beragam bentuk.

2.1.5. HTTP

HTTP merupakan singkatan dari *Hyper Text Transfer Protocol* yaitu protokol yang digunakan untuk mentransfer sebuah halaman *World Wide Web* dari satu komputer ke komputer lainnya (Douglas E. Comer, 2004, p656). Kebanyakan *browser* menganggap bahwa semua alamat web pasti dimulai dengan *http://*, sehingga anda tidak perlu mengetik bagian tersebut, cukup ketik bagian sisanya yang dimulai dengan www.

2.1.6. Teori Dasar SMS

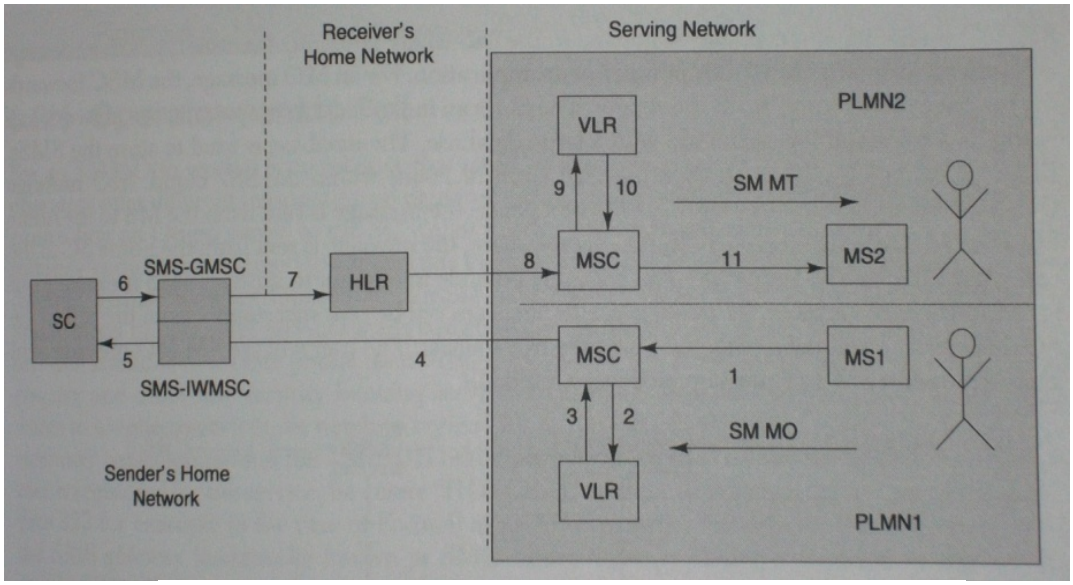
2.1.6.1. SMS

Menurut Puneet Gupta, SMS (Short Message Service) adalah sebuah mekanisme pengiriman pesan singkat melalui jaringan selular. Pesan singkat dari ponsel pengirim akan disimpan di SMS Center (SMSC) yang kemudian akan meneruskan pesan ke ponsel penerima. Pada tanggal 3 Desember 1992, seorang ilmuwan dari Sema, perusahaan teknologi Inggris, bernama Neil Papworth mengirim pesan singkat pertama yang berisi ucapan selamat natal ke direktur Vodafone, Richard Jarvis. Pesan ini dikirim lebih cepat sebelum natal, sekitar 3 minggu sebelumnya. Vodafone menawarkan layanan pesan singkat ini dengan nama TeleNotes service yang ditargetkan untuk komunitas bisnis. Tapi layanan ini tidak populer sama sekali pada awalnya. SMS hampir dilupakan sampai setelah 7 tahun berikutnya yaitu tahun 1999 ketika salah satu operator ponsel memulai mengijinkan pelanggan untuk bertukar SMS. Saat ini SMS menjadi layanan pengiriman pesan yang paling populer dengan rata-rata satu milyar pesan dikirimkan setiap hari di seluruh dunia.

Data yang dapat ditangani oleh SMS pada dasarnya sangat terbatas. Sebuah SMS hanya bisa berisikan paling banyak 160 karakter jika menggunakan 7-bit karakter Inggris, 140 karakter jika menggunakan 8-bit karakter (beberapa alfabet Eropa) dan 70

karakter jika menggunakan alfabet non-latin seperti alfabet Arab, China, atau Jepang (70 karakter dari unicode 16 bit).

2.1.6.2. Arsitektur SMS



Gambar 2. 1 Arsitektur SMS

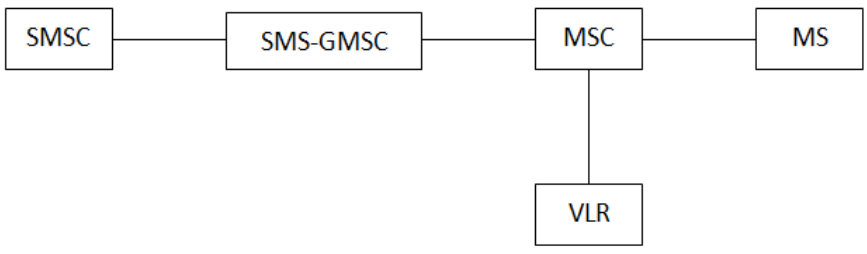
- MS (Mobile Station) adalah terminal wireless yang mampu menerima dan membuat pesan singkat serta panggilan suara.
- SME (Short Messaging Entities) adalah sebuah entitas yang bertugas mengirim dan menerima pesan dan dapat ditemukan di jaringan yang fixed atau sebuah mobile station.
- SMSC (Short Message Service Center) bertanggung jawab terhadap penyimpanan dan penyampaian pesan singkat antara sebuah SME dengan MS (Mobile Station).

- SMS Gateway MSC (SMS-GMSC) adalah sebuah MSC yang mampu menerima pesan singkat dari SMSC, meminta informasi routing pada HLR, dan mengirimkan pesan singkat ke MSC MS penerima. SMS Interworking MSC (SMS-IWSMC) adalah sebuah MSC yang mampu menerima pesan singkat dari jaringan selular dan mengirimkannya ke SMSC yang sesuai. SMS-GMSC/SMS-IWMSC biasanya diintegrasikan dengan SMSC.
- HLR (Home Location Register) adalah sebuah database yang digunakan untuk penyimpanan permanen dan pengelolaan subscription dan layanan profil. Ketika informasi diminta oleh SMSC, HLR menyediakan informasi routing untuk subscriber yang ditunjuk. HLR juga menginformasikan SMSC, yang sebelumnya pesan singkat gagal dikirimkan ke MS tertentu, bahwa MS yang sekarang dikenali oleh jaringan selular dapat diakses.
- MSC (Mobile Switching Center) melakukan fungsi switching sistem dan kontrol panggilan ke dan dari sistem telepon lainnya.
- VLR (Visitor Location Register) adalah database yang mengandung informasi sementara tentang subscriber. Informasi ini dibutuhkan oleh MSC untuk melayani subscriber yang masuk.

Pada dasarnya SMS terbagi menjadi dua jenis, SM MT (Short Message Mobile Terminated Point-to-Point) dan SM MO (Short Message Mobile Originated Point-to-Point). SM MT adalah pesan masuk dari sisi jaringan dan diakhiri didalam MS. MS MO adalah pesan keluar yang berasal dari MS dan diteruskan ke jaringan untuk dikirimkan. Untuk pesan keluar, jalur dari MS ke SMSC melalui VLR dan IWMSC, sedangkan jalur pesan masuk adalah dari SMSC ke MS melalui HLR dan GMSC.

2.1.6.3. Short Message Mobile Terminated (SM MT)

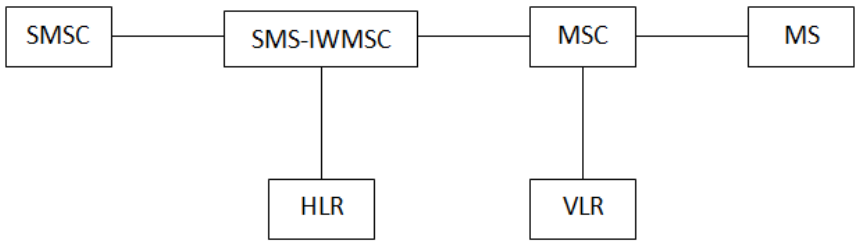
Untuk pesan SM MT, pesan dikirim dari SMSC ke MS. Seluruh proses ini dilakukan didalam satu transaksi. Untuk mengirimkan pesan masuk, SMSC menyediakan jaringan yang tidak pernah digunakan. Hal ini menyatakan bahwa pesan SMS dapat dikirim dari SMSC mana saja di jaringan mana saja pada ponsel GSM manapun di dunia.



Gambar 2. 2 *Short Message Mobile Terminated* (SM MT)

2.1.6.4. Short Message Mobile Originated (SM MO)

SM MO adalah pesan keluar yang berasal dari MS dimana pada umumnya pengguna mengetik pesan dan mengirimkannya ke nomor MSISDN (Mobile Subscriber ISDN Number) yang lain atau ke aplikasi. Untuk pesan MO, MSC meneruskan pesan ke SMSC pengirim. SMSC tidak bergantung pada komputer di dalam jaringan dan bekerja sebagai penyimpan dan meneruskan node dengan database yang besar. Database digunakan untuk menyimpan SMS. Dalam terminologi SS7, SMSC adalah Service Control Point (SCP) dengan SS7 cloud. Pesan MO bekerja pada dua fase asynchronous. Pada fase pertama, pesan dikirimkan ke MS dari SMSC sebagai sebuah pesan MO. Dalam fase kedua, pesan dikirimkan dari SMSC ke MS penerima sebagai sebuah pesan MT. Hal tersebut memungkinkan pesan dikirim ke nomor MSISDN yang invalid. Dalam kasus ini, pesan akan berhasil dikirimkan dari MS ke SMSC. Namun, pesan akan gagal ketika SMSC mengirim ke MS. Pengguna akan melihat pesan SM MO berhasil dikirimkan tetapi pesan SM MT akan gagal untuk dikirim.



Gambar 2. 3 *Short Message Mobile Originated* (SM MO)

2.1.6.5. Keunggulan SMS

- ✓ Omnibus nature of SMS, SMS menggunakan saluran sinyal SS7 yang tersedia diseluruh dunia. SMS adalah satu-satunya bearer yang memungkinkan subscriber mengirim SMS jarak jauh tanpa memiliki subscription jarak jauh. Sebagai contoh, anda tidak dapat melakukan panggilan suara ke ponsel di Inggris kecuali anda memiliki fasilitas panggilan internasional. Namun, anda dapat mengirim SMS ke subscriber di Inggris tanpa memiliki fasilitas panggilan internasional.
- ✓ Stateless, sifat SMS adalah sessionless dan stateless. Setiap SMS searah dan tidak bergantung pada konteks apapun. Hal ini membuat SMS menjadi bearer terbaik untuk notifikasi, alert dan paging.
- ✓ Asynchronous, ketika penerima sedang out of service, pengiriman tidak akan dibiarkan begitu saja. Karena itu, SMS dapat digunakan sebagai antrian pesan. Pada dasarnya, SMS dapat digunakan untuk membawa bearer pada kedua pertukaran informasi yaitu synchronous dan asynchronous.
- ✓ Self-configurable and last mile problem resistant, pada kasus Web, tidak ada cara untuk terhubung ke layanan dari network asing tanpa melakukan perubahan konfigurasi apapun. Perangkat perlu dikonfigurasi oleh user atau system administrator untuk mengakses ke jaringan. Ketika di

jaringan asing, ponsel dapat mengakses SMS bearer tanpa melakukan perubahan apapun pada pengaturan ponsel. Subscriber selalu dihubungkan ke SMS bearer terlepas dari konfigurasi apapun. Ketika memasuki ke jaringan asing, bahkan jika penyedia jaringan tidak memiliki SMSC, SMS tetap dapat dikirim dan diterima.

- ✓ Non-repudiable, pesan SMS membawa SMSC dan sumber MSISDN sebagai bagian dari header pesan. Tidak seperti IP address, SMS tidak mudah ditentukan alamat MSISDN dalam SMS. SMS memungkinkan aplikasi dihubungkan ke SMS untuk menentukan alamat MSISDN seperti “999”. Namun, aplikasi tidak dapat menentukan alamat dari SMSC. Karena itu, SMS dapat membuktikan keasliannya sendiri tanpa diragukan.
- ✓ Always Connected, karena SMS menggunakan saluran sinyal SS7 untuk trafik datanya, media bearer selalu aktif. Pengguna tidak dapat mematikan, melarang atau mengalihkan pesan SMS manapun. Ketika ponsel sedang sibuk atau sedang ada panggilan, pesan SMS tetap dikirimkan ke MS tanpa harus memutuskan panggilan.

2.1.7. Modem

Menurut Williams, (2003, p32), modem (modulator/demodulator) adalah sebuah perangkat yang mampu mengirim dan menerima data melalui kabel telepon dari dan ke komputer. Modem pengirim akan memodulasi sinyal digital ke dalam bentuk sinyal analog agar bisa ditransmisikan melalui kabel telepon. Modem penerima akan men-demodulasi sinyal analog kembali ke dalam bentuk sinyal digital.

2.2 Teori-teori Khusus

2.2.1. *Unified Modeling Language* (UML)

Menurut Tom Pender (UML Bible © 2003, chapter 1), *Unified Modeling Language* adalah sejumlah teknik pemodelan yang diambil dari metodologi yang beragam yang telah dipraktekkan selama dua dekade sebelumnya. Sejak mulai dikembangkannya pada tahun 1994, saat ini UML telah menjadi standar *de facto* untuk membuat model perangkat lunak berbasiskan object.

UML memungkinkan pengembang sistem untuk menentukan, memvisualisasikan, dan mendokumentasikan model dengan cara yang mendukung skalabilitas, keamanan, dan eksekusi yang kuat. Selain itu UML juga dirancang untuk memenuhi beberapa tujuan yang sangat spesifik sehingga benar-benar menjadi standar yang membahas kebutuhan praktis komunitas pengembangan perangkat lunak.

Tujuan-tujuan UML :

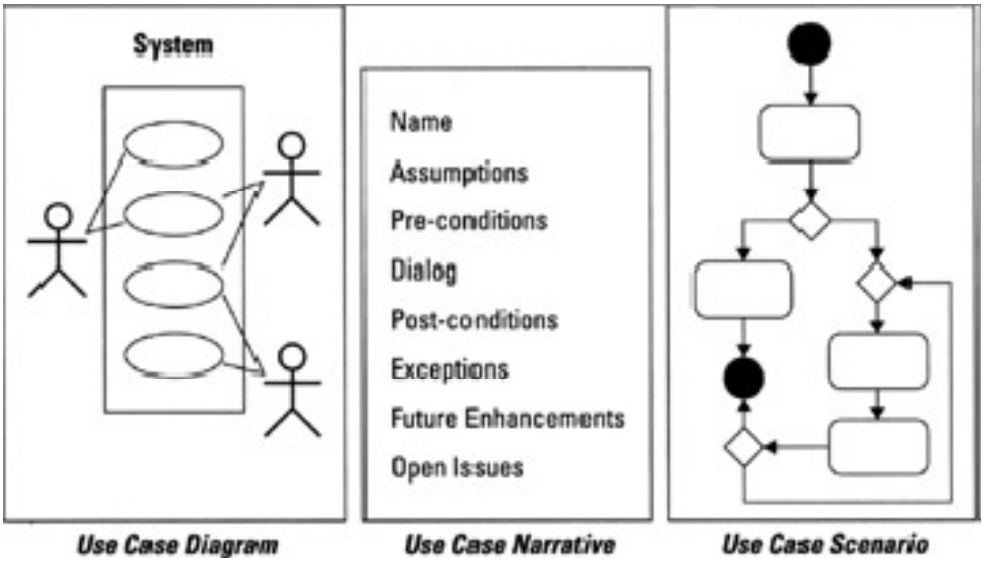
1. Menyediakan model yang siap digunakan, ekspresif, dan bahasa pemodelan visual untuk mengembangkan dan bertukar model.
2. Memberikan mekanisme yang bisa diperpanjang dan terspesialisasi untuk meningkatkan konsep dari inti yang ada.
3. Dukungan spesifikasi yang independen terhadap bahasa pemrograman tertentu dan pengembangan proses.
4. Memberikan dasar formal untuk memahami bahasa pemodelan.
5. Mendukung konsep pengembangan yang lebih tinggi tingkat seperti komponen, kolaborasi, kerangka kerja (*frameworks*), dan pola (*patterns*).

Beberapa diagram yang terdapat dalam UML :

- *Use Case Diagram* : Diagram menggambarkan siapa yang akan menjadi pengguna yang relevan, layanan yang mereka butuhkan dari sistem, dan layanan yang mereka butuhkan untuk menjalankan sistem.
- *Activity Diagram* : Diagram yang merupakan pengembangan dari flowchart, dimana diagram ini menunjukkan urutan-urutan tugas yang dilakukan oleh tiap-tiap titik.
- *Class Diagram* : Diagram yang menjelaskan tentang definisi dan kegunaan dari object-object yang ada.
- *Sequence Diagram* : Diagram yang menjelaskan interaksi antar object dari waktu ke waktu.

2.2.1.1. *Use Case Diagram*

Use Case Diagram adalah elemen grafis yang unik, dalam hal ini adalah diagram yang digunakan untuk memodelkan bagaimana orang memperkirakan bagimana cara untuk menggunakan sistem. Diagram ini menggambarkan siapa yang akan menjadi pengguna yang relevan, layanan yang mereka butuhkan dari sistem, dan layanan yang mereka butuhkan untuk menjalankan sistem. (**Tom Pender,** UML Bible © 2003, c12)



Gambar 2. 4 Use Case

Elemen-elemen dalam *use case* diagram :

- *Actor* : Sebuah peran yang dimainkan oleh seseorang, sistem, perangkat, atau bahkan perusahaan, yang memiliki peran dalam sistem.
- *Use case* : Mengidentifikasi perilaku dari sistem. Tanpa perilaku ini, sistem tidak akan memenuhi persyaratan actor. Setiap use case mengungkapkan tujuan yang harus dicapai oleh sistem harus dan / atau hasil yang harus dihasilkan.

- *Association* : Mengidentifikasi interaksi antara *actor* dan *use case*. Setiap *association* menjadi dialog yang harus dijelaskan dalam sebuah narasi *use case*. Setiap narasi pada gilirannya menyediakan satu set skenario yang dapat membantu dalam pengembangan dari kasus uji ketika mengevaluasi analisis, desain dan implementasi artefak dari *use case* dan *association*.
- Include Relationship : Mengidentifikasi use case yang dapat digunakan kembali, yang dapat dimasukkan ke dalam pelaksanaan use case lainnya.
- Extend Relationsip : Mengidentifikasi use case yang dapat digunakan kembali yang mengganggu pelaksanaan use case lainnya.
- Generalization : Mengidentifikasi hubungan inheritance antara actor atau antara use case.

Langkah-langkah dalam membuat use case :

1. Mengidentifikasi konteks dari sistem :
   1) Mengidentifikasi actor dan tanggung jawab mereka.
   2) Mengidentifikasi use case, perilaku dari sistem, dalam tujuan tertentu dan / atau hasil yang harus diproduksi.
2. Mengevaluasi actor dan use case untuk menemukan peluang untuk perbaikan, seperti membelah atau penggabungan definisi
3. Mengevaluasi use case untuk menemukan «include» relationship.

4. Mengevaluasi use case untuk menemukan «extend» relationship.
5. Mengevaluasi actor dan use case untuk menemukan generalization.

2.2.1.2. *Activity Diagram*

Menurut Tom Pender (UML Bible © 2003, chapter 13), Activity Diagram adalah diagram yang merupakan pengembangan dari flowchart, dimana diagram ini menunjukkan urutan-urutan tugas yang dilakukan oleh tiap-tiap titik.

Notasi activity diagram :

- Kegiatan dan transisi

  Kegiatan merupakan langkah dalam proses di mana beberapa pekerjaan selesai dilakukan. Ini bisa berupa perhitungan, menemukan beberapa data, memanipulasi informasi, atau memverifikasi. Kegiatan diwakili oleh persegi panjang bulat yang mengandung teks bebas.

Gambar 2. 5 Activity Diagram Kegiatan dan Transisi

- Penjaga kondisi

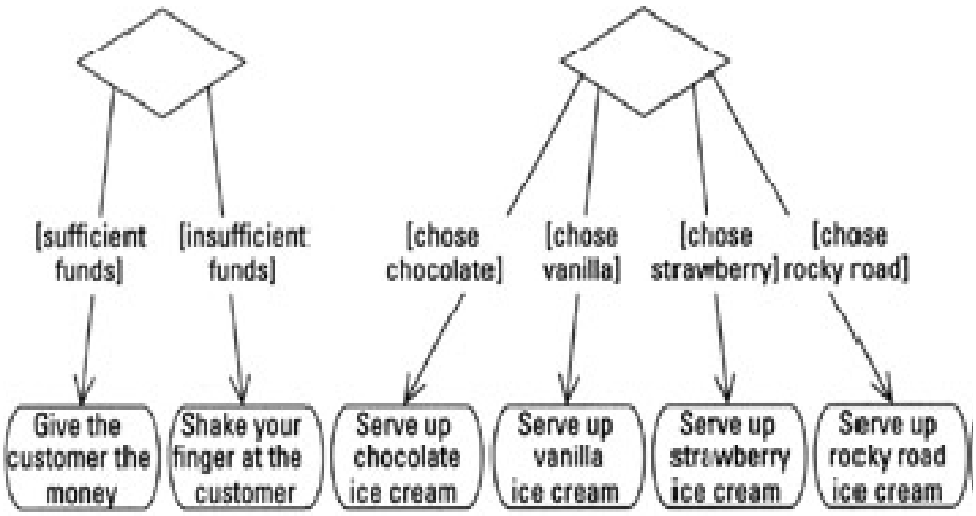
  Kadang-kadang transisi harus digunakan hanya ketika hal-hal tertentu terjadi. Penjaga kondisi dapat ditugaskan untuk

transisi untuk membatasi penggunaannya. Tempatkan kondisi dalam tanda kurung siku di suatu tempat dekat transisi panah. Kondisi ini harus menguji kebenarannya sebelum dapat mengikuti transisi terkait berikutnya.

Gambar 2. 6 Activity Diagram Penjaga Kondisi

- Keputusan

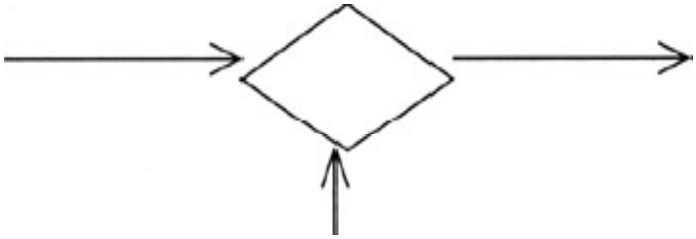
  Activity diagram berlian adalah ikon keputusan, sama seperti di flowchat. Satu panah keluar dari berlian untuk setiap nilai yang mungkin dari kondisi diuji. Keputusan harus merupakan keputusan yang sederhana seperti tes benar / salah.



Gambar 2. 7 Activity Diagram Keputusan

- Gabung titik

  Ikon berlian juga digunakan untuk model dari penggabungan titik, tempat di mana dua jalur alternatif yang datang bersama-sama dan melanjutkan sebagai salah satu.



Gambar 2. 8 Activity Diagram Gabung Titik

- Mulai dan akhir

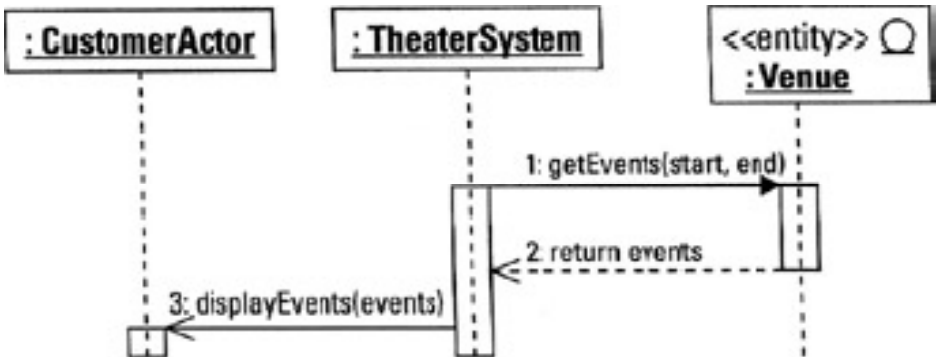
  UML juga menyediakan ikon untuk memulai dan mengakhiri suatu diagram Kegiatan.

Gambar 2. 9 Activity Diagram Mulai dan Akhir

2.2.1.3. *Sequence Diagram*

Sequence diagram adalah diagram yang menjelaskan interaksi antar *object* dari waktu ke waktu. Manfaat utama dari diagram ini adalah untuk membantu mengidentifikasi pesan yang dipertukarkan antar *object*. Untuk bertukar pesan, dibutuhkan pengirim dan penerima. Sebuah penerima harus memiliki sebuah antarmuka untuk menerima pesan. Oleh karena itu, jika pesan harus dikirim dari satu objek ke yang lain, penerima harus

mendefinisikan antarmuka yang sesuai dengan pesan. Sebuah antarmuka adalah sebuah ciri khas operasi pada class yang dimiliki oleh objek penerima. *Sequence diagram* membantu kita menemukan dan mendokumentasikan operasi baru di-*class diagram*. (**Tom Pender,** UML Bible © 2003, chapter 3)



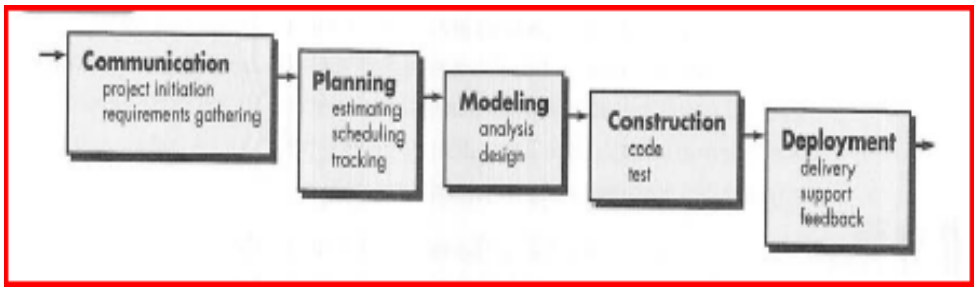
Gambar 2. 10 Sequence Diagram

2.2.2. Model-model Proses Perangkat Lunak

(Roger S. Pressman, Ph.D., Software Engineering A Practitioner's Approach Seventh Edition © 2010, chapter 2)

2.2.2.1. Model Sekuensial linier / *Waterfall*

Model sekuensial linier merupakan model yang paling sering digunakan karena lebih mudah dimengerti.



Gambar 2. 11 Waterfall

Model sekuensial meliki tahapan-tahapan sebagai berikut:

- Analisis kebutuhan perangkat lunak

  Melakukan pengumpulan kebutuhan, khususnya perangkat lunak untuk memahami sifat program yang dibangun. Analis harus memahami *domain* informasi, tingkah laku, unjuk kerja, dan antarmuka yang diperlukan. Kebutuhan baik untuk sistem maupun untuk perangkat lunak didokumentasikan dan dilihat lagi bersama dengan pelanggan.

- *Desain*

  Proses *desain* perangkat lunak adalah proses multi langkah yang berfokus pada empat atribut sebuah program yang berbeda; struktur data, arsitektural perangkat lunak, representasi interface, dan *detail* (algoritma) *prosedural*.

  Proses desain menerjemahkan syarat/kebutuhan ke dalam sebuah representasi perangkat lunak yang dapat diperkirakan demi kualitas sebelum dimulai pemunculan kode. Sebagaimana persyaratan, desain didokumentasikan dan menjadi bagian dari konfigurasi perangkat lunak.

- Generasi kode

  Desain diterjemahkan ke dalam bentuk yang dapat dibaca oleh mesin. Tahap generasi kode melakukan tugas ini. Jika desain dilakukan secara lengkap, pembuatan kode dapat diselesaikan secara sistematis.

- Pengujian

  Setelah kode selesai dibuat, pengujian program dimulai. Proses pengujian berfokus pada logika internal perangkat lunak, memastikan bahwa semua pernyataan sudah diuji, dan data eksternal fungsional, yaitu mengarahkan pengujian untuk menemukan kesalahan-kesalahan dan memastikan bahwa *input* uang dibatasi akan memberikan hasi aktual yang sesuai dengan hasil yang dibutuhkan.

- Pemeliharaan

  Perangkat lunak akan mengalami perubahan setelah disampaikan kepada pelanggan. Perubahan akan terjadi karena kesalahan-kesalahan ditemukan, karena perangkat lunak harus disesuaikan untuk mengakomodasi perubahan-perubahan di dalam lingkungan eksternalnya, atau karena pelaangan membutuhkan perkembangan fungsional atau unjuk kerja. Pemeliharaan perangkat lunak mengaplikasikan lagi setiap fase program sebelumnya dan tidak membuat yang baru lagi.

Model ini memiliki beberapa kelemahan, yaitu :

- ✓ Jarang sekali proyek nyata mengikuti aliran sekuensial yang dianjurkan oleh model.
- ✓ Kadang-kadang sulit bagi pelanggan untuk menyatakan semua kebutuhannya secara eksplisit. Model linier sekuensial memerlukan hal ini dan mengalami kesulitan untuk

mengakomodasi ketidakpastian *natural* yang ada pada bagia neberapa proyek.

- ✓ Pelanggan harsus bersikap sabar. Sebuah versi kerja dari program-program itu tidak akan diperoleh sampai akhir waktu proyek dilalui. Sebuah kesalahan besar, jika tidak terdeteksi sampai program yang bekerja tersebut dikasji ulang, bisa menjadi petaka.
- ✓ Pengembang sering melakukan penundaan yang tidak perlu. Di dalam analisis yang menarik tentang proyek aktualm Bradac [BRA94] mendapatkan bahwa sifat alami dari siklus kehidupan klasik membawa kepada blocking state di mana banyak anggota tim proyek harus menunggu tim yang lain untuk melengkapi tugas yang saling memiliki ketergantungan. Kenyataannya, waktu yang dipake untuk menunggu bisa mengurangi waktu untuk usaha produktif

2.2.2.2. Model Inkremental

Model inkremental menggabungkan elemen dari aliran proses sekuensial linear dan paralel. Ada banyak situasi di mana persyaratan perangkat lunak awal yang cukup baik didefinisikan, tetapi lingkup keseluruhan upaya pembangunan menghalangi proses linear. Selain itu, mungkin ada kebutuhan mendesak untuk menyediakan satu set fungsi terbatas perangkat lunak untuk pengguna dengan cepat dan kemudian memperbaiki dan

memperluas fungsionalitas perangkat lunak yang di rilis nanti. Dalam kasus seperti itu Anda dapat memilih model proses yang dirancang untuk menghasilkan perangkat lunak secara bertahap.

2.2.2.3. Model Prototipe

Dimulai dengan pengumpulan kebutuhan. Pegembang dan pelanggan bertemu dan mendefinisikan objektif keseluruhan dari perangkat lunak, mengidentifikasi segala kebutuhan yang diketahui, dan area garis besar di mana definisi lebih jauh merupakan keharusan kemudian dilakukan “perancangan kilat”. Perancangan kilat berfokus pada penyajian dari aspek-aspek perangkat lunak tersebut yang nampak bagi pelanggan/pemakai (contohnya pendekatan *input* dan *format output*). Perancangan kilat membawa kepada konstruksi sebuah prototipe. Prototipe tersebut dievaluasi oleh pelanggan/pemakai dan dipakai untuk menyaring kebutuhan pengembangan perangkat lunak. Iterasi terjadi pada saat yang sama memungkinkan pengembang untuk secara lebih baik memahami apa yang harus dilakukannya.

Masalah-masalah yang mungkin muncul saat menggunakan model prototipe :

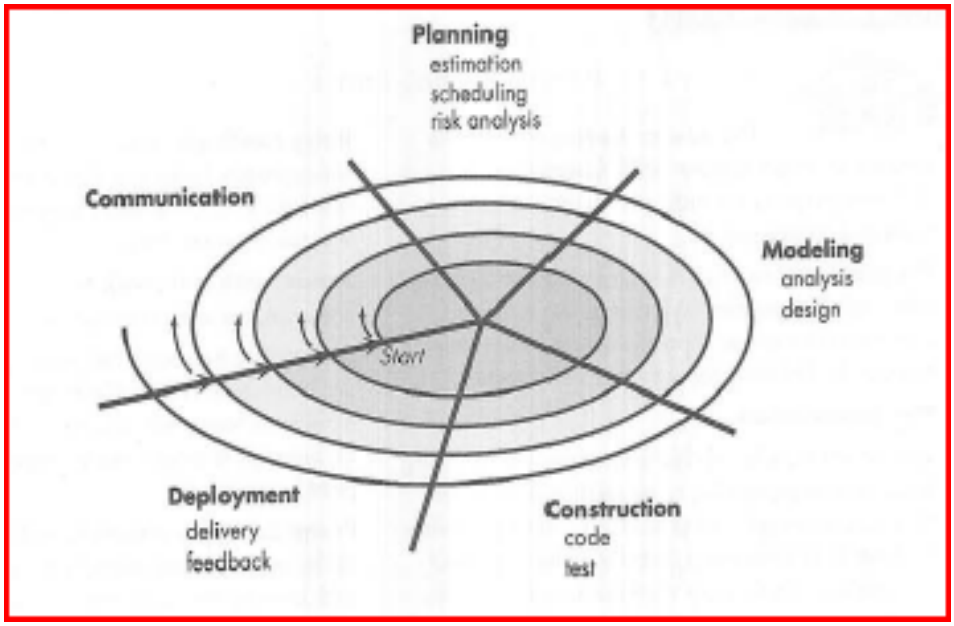
- Pelanggan melihat apa yang tampak sebagai versi perangkat lunak yang bekerja tanpa melihat bahwa prototipe itu dijalin bersama-sama “dengan permen karet dan baling wire”, tanpa melihat bahwa di dalam permintaan untuk membuatnya

bekerja, kita belum mencantumkan kualitasi perangkat lunak secara keseluruhan atau kemampuan pemeliharaan untuk jangkja waktu yang panjang. Ketika diberi informasi bahwa produk harus dibangun lagi agar tingkat kualitas yang tinggi bisa dijaga, pelanggan akan meneriakkan kecurangan dan permintaan agar dipakai “beberapa campuran” untuk membuat prototipe menjadi sebuah produk yang bekerja yang lebih sering terjadi, sehingga manajemen pengembangan perangkat lunak menjadi penuh dengan belas kasihan.

- Pengembang sering membuat kompromi-kompromi implementasi untuk membuat prototipe bekerja dengan cepat. Sistem oprasi atau bahasa pemrograman yang tidak sesuai bisa dipakai secara sederhana karena mungkin lemih muda diperoleh dan dikenal; algoritma yang tidak efisien secara sederhana bisa diimplementasikan untuk mendemonstrasikan kemampuan. Setelah beberapa waktu, pengembang mungkin lebih memilih pilihan yang telah digunakan tersebut dan melupakan semua alasan mengapa mereka tidak cocok dengan algoritma tersebut. Pilihan yang kurang ideal itu telah menjadi bagian integral dari sistem tersebut.

2.2.2.4. Model *Spiral*

Model *spiral* adalah proses perangkat lunak yang evolusioner yang merangkai sifat iteratif dari prototipe dengan cara kontrol dan aspek sistematis dari model sekuensial linier. Model ini berpeotensi untuk pengembangan versi tambahan secara cepat. Dalam model ini, perangkat lunak dikembangkan di dalam suatu deretan pertambahan. Selama awal iterasi, rilis inkremental bisa merupakan sebuah model atau prototipe kertas. Selama iterasi berikutnya, sedikit demi sedikit dihasilkan versi sistem rekayasa yang lebih lengkap.



Gambar 2. 12 Spiral

Tahapan-tahapan :

- ✓ Komunikasi : Melakukan komunikasi dengan pelanggan untuk mengetahui sistem seperti apa yang dibutuhkan.

- ✓ Perencanaan : Mendefinisikan sumber daya, waktu, dan informasi-informasi lain yang dibutuhkan.
- ✓ Analisa : Menganalisa resiko apa saja yang ada, baik manajemen mapun teknis.
- ✓ Perekayasaan : Membangun satu atau lebih representasi dari aplikasi tersebut.
- ✓ Konstruksi dan peluncuran : Mengkonstruksi, menguji, memasang, dan memberikan pelayanan terhadap pemakai seperti pelatihan dan dokumentasi.
- ✓ Evaluasi : Memperoleh umpan balik dari pelanggan dengan didasarkan pada evaluasi representasi perangkat lunak, yang dibuat selama pasa perekayasaan, dan implementasikan selama masa pemasangan.

2.2.2.5. *Agile*

Model agile merupakan salah satu model untuk membuat perangkat lunak secara cepat. Setiap proses perangkat lunak agile memiliki ciri yang membahas sejumlah asumsi kunci tentang mayoritas proyek perangkat lunak, yaitu:

1. Sulit untuk memprediksi di muka, mana persyaratan perangkat lunak yang akan bertahan dan mana yang akan berubah. Hal ini sama sulitnya dengan memprediksi bagaimana prioritas pelanggan akan berubah seiring dengan proses proyek .

2. Untuk berbagai jenis desain dan konstruksi perangkat lunak desain dan konstruksi yang ditinggalkan. Kedua aktivitas harus dilakukan bersama-sama sehingga model desain terbukti ketika mereka diciptakan. Sulit untuk memperkirakan berapa banyak desain diperlukan sebelum konstruksi digunakan untuk membuktikan desain.
3. Analisis, desain, konstruksi, dan pengujian yang tidak dapat diprediksi (dari perencanaan sudut pandang) seperti yang kita inginkan.

Mengingat tiga asumsi diatas, sebuah pertanyaan penting muncul: Bagaimana kita menciptakan sebuah proses yang dapat mengelola ketidakpastian? Jawabannya, terletak pada proses adaptasi (untuk proyek yang cepat berubah dan kondisi teknis). Oleh karena itu, sebuah proses agile harus bisa beradaptasi. Tapi adaptasi terus menerus tanpa kemajuan ke depan hanya menyelesaikan sedikit masalah. Oleh karena itu, proses perangkat lunak agile harus beradaptasi secara bertahap Untuk mencapai adaptasi incremental, tim agile membutuhkan umpan balik pelanggan (sehingga adaptasi yang tepat dapat dibuat). Katalis yang efektif untuk umpan balik pelanggan adalah pototype operasional atau sebagian dari system. Oleh karena itu, strategi operasional pengembangan inkremental harus dilembagakan. Peningkatan perangkat lunak (prototipe yang bisa dieksekusi atau bagian dari sistem operasional) harus disampaikan dalam jangka

waktu pendek sehingga adaptasi tetap memiliki ruang dengan perubahan (ketidakpastian). Pendekatan iteratif memungkinkan pelanggan untuk mengevaluasi kenaikan perangkat lunak secara teratur, memberikan umpan balik yang diperlukan untuk tim perangkat lunak, dan pengaruh proses adaptasi yang dibuat untuk mengakomodasi umpan balik.

Aliansi Agile mendefinisikan 12 prinsip Agile bagi mereka yang ingin mencapai kecepatan:

1) prioritas tertinggi kita adalah untuk memuaskan pelanggan melalui pengiriman awal dan terus menerus dari perangkat lunak yang berarti.
2) menyambut perubahan kebutuhan, bahkan di tahap akhir pembangunan. Proses agile memanfaatkan perubahan sebagai keunggulan kompetitif pelanggan.
3) Menyediakan perangkat lunak yang berjalan secara terus menerus, dari beberapa minggu sampai beberapa bulan, dengan preferensi dengan skala waktu yang lebih pendek.
4) Orang bisnis dan pengembang harus bekerja sama setiap hari sepanjang proyek.
5) Membangun proyek di antara individu-individu yang termotivasi. Beri mereka lingkungan dan dukungan yang mereka butuhkan, dan berikan kepercayaan kepada mereka untuk menyelesaikan pekerjaan yang ada.

6) Metode yang paling efisien dan efektif untuk menyampaikan informasi ke dan dalam tim pengembangan adalah percakapan tatap muka.
7) Perangkat lunak yang berfungsi adalah alat ukur utama dalam menentukan kemajuan.
8) Proses agile mempromosikan pembangunan berkelanjutan. Para sponsor, pengembang, dan pengguna harus mampu mempertahankan kecepatan konstan tanpa batas.
9) Perhatian terus menerus untuk keunggulan teknis dan desain yang baik meningkatkan kelincahan
10) kesederhanaan - seni memaksimalkan jumlah pekerjaan tidak dilakukan adalah penting.
11) Arsitektur terbaik, persyaratan, dan desain muncul dari mengorganisir diri tim.
12) Pada interval reguler, tim mencerminkan tentang bagaimana untuk menjadi lebih efektif dan kemudian menyesuaikan perilakunya.

2.2.3. PHP

Menurut Dodit Suprianto (2008, p17), PHP merupakan kependekan dari *Hypertext Preprocessor*. PHP adalah bahasa scripting yang sangat cocok untuk pengembangan Web dan dapat tertanam ke dalam HTML. PHP tergolong sebagai bahasa pemrograman yang berbasis *server* (*Server Side Scripting*). PHP mempunyai banyak

keunggulan dibandingkan dengan bahasa pemrograman berbasis *web* yang lain, seperti *open source software,* kesederhanaan bahasa PHP, PHP yang mampu berintegrasi dengan berbagai macam jenis *database. Database* yang paling umum digunakan dalam PHP adalah MySQL.

2.2.4. Jquery

Menurut jquery.com, Jquery adalah library JavaScript yang cepat dan ringkas yang menyederhanakan dokumen html, penanganan *event*, animasi dan interaksi ajax untuk pengembangan web cepat.

2.2.5. *Cascading Style Sheet (CSS)*

Menurut Elizabeth Castro, (2006, chapter 7)Css adalah sebuah *file text* sederhana yang mengandung satu atau lebih aturan yang menentukan melalui properti dan unsur-unsur tertentu dalam halaman web yang harus ditampilkan. Ada properti CSS untuk mengendalikan format dasar seperti ukuran font dan warna, tata letak properti seperti posisi dan mengambang, dan kontrol cetak untuk memutuskan tempat istirahat halaman akan muncul. CSS juga memiliki sejumlah sifat dinamis, yang memungkinkan item untuk muncul dan menghilang dan berguna untuk membuat menu *drop down* dan komponen interaktif lainnya.

2.2.6. Gammu

Gammu adalah sebuah nama proyek serta nama dari sebuah utilitas berbasiskan *command line,* dimana kita dapat menggunakannya

untuk mengontrol ponsel. Utilitas ini ditulis dengan bahasa C dan dibangun dengan menggunakan *library* libGammu.

Proyek ini telah berasal dari *Gnokii* pada awalnya dan sampai versi 0.58 telah bernama *MyGnokii2*. Kebutuhan akan nama yang lebih baik mendorong keluarnya nama Gammu yang mempunyai akronim yaitu Gammu All Mobile Management Utilities, tanpa mengetehui bahwa kata Gammu itu sudah digunakan dari buku "*Bidat Dune*" yang ditulis oleh Frank Herbert.

Gammu merupakan *software* yang bersifat *open source* yang digunakan sebagai *tool* untuk mengembangkan aplikasi SMS *Gateway*, cukup mudah diimplementasikan, dan tidak berbayar. Kelebihan Gammu dari *tool* SMS *gateway* lainnya adalah:

- Gammu dapat dijalankan di sistem operasi Linux maupun Windows.
- Banyak *device* yang kompatibel di Gammu.
- Gammu menggunakan *database* MySQL untuk menyimpan SMS yang ada pada kotak masuk (*inbox*) maupun untuk mengirim pesan, sehingga dapat dibuat *interface* yang berbasis web maupun desktop.
- Baik kabel data USB maupun serial, semuanya kompatibel di Gammu.

Paket Gammu sudah termasuk *binary file*, *SMS Daemon*, *Gammy Library*, dan *Phyton Bindings* yang dapat anda gunakan untuk mengembangkan aplikasi anda yang dapat mengakses ponsel.