BAB 2

LANDASAN TEORI

Bab ini berisikan teori-teori pendukung yang digunakan dalam pembuatan sistem ini, berikut teori-teori yang digunakan :

2.1. Teori Umum

2.1.1. Basis Data

2.1.1.1. Pengertian Data

Data merupakan suatu objek, kejadian, atau fakta yang terdokumentasikan dengan memiliki kodifikasi terstruktur untuk suatu atau beberapa entitas. Data tersebut akan menghasilkan informasi. Jenis data berdasarkan sumber Menurut Rabianski (2003) :

1. Data Primer

   Data primer adalah secara langsung diambil dari objek penelitian oleh peneliti perorangan maupun organisasi.

   Contoh : Wawancara langsung penonton bioskop 21 untuk meneliti preferensi konsumen bioskop.

2. Data Sekunder

   Data sekunder adalah data yang didapat tidak secara langsung dari objek penelitian.

Peneliti mendapatkan data yang sudah jadi yang dikumpulkan oleh pihak lain dengan berbagai cara atau metode baik secara komersial maupun non komersial.

Contoh : pada peneliti yang menggunakan data statistik hasil riset dari surat kabar atau majalah.

2.1.1.2. Pengertian *Database*

Menurut Connoly T. M. dan Begg C. E. (2006:15) *Database* adalah sekumpulan data yang saling berhubungan dan dapat mendeskripiskan suatu data untuk dirancang menjadi suatu informasi yang dibutuhkan suatu organisasi. *Database* harus memiliki integrasi antar data dengan tingkat duplikasi data yang minimum untuk mencegah *redundant data*.

Menurut Ramakrishnan, Gehrke, Raghu, et.al (2005:4) *Database* adalah kumpulan data yang mendeskripsikan aktifitas antar bagian di dalam suatu organisasi.

2.1.1.3. Pengertian *Database Relational*

Database Relational adalah suatu model *database* yang disajikan dalam bentuk tabel. Model ini diperkenalkan pertama kali oleh E.F. Codd pada tahun 1970. Sebuah *database relational* terdiri dari koleksi tabel-tabel, yang masing-masing diberikan dengan nama yang unik. Sebuah baris dalam tabel merepresentasikan sebuah keterhubungan atau *relationship* dari beberapa nilai yang ada.

2.1.1.4. *Database Management System*

Menurut Connoly T. M. dan Begg C. E (2006:16) *Database Management System* (DBMS) adalah suatu *software* yang merancang, membuat, mengatur dan mengendalikan seluruh proses di dalam *database*.

Menurut Ramakrishnan, Gehrke, Raghu, et.al. (2005:4) DBMS merupakan *software* yang dirancang untuk mengatur dan menggabungkan data dalam jumlah besar.

DBMS memberikan fasilitas kepada *user* untuk mendefinisikan suatu *database* melalui proses *Data Definition Language* (DDL) dan dapat melakukan *insert, update, delete* dan mendapatkan kembali data dari *database* dengan melakukan proses *Data Manipulation Language* (DML).

DBMS memiliki beberapa keuntungan dan kerugian, yaitu :

A. Keuntungan

1. Mengendalikan *redundancy data*
2. Konsistensi data
3. Meningkatkan integrasi data
4. Meningkatkan keamanan
5. Mengambil data dan informasi secara cepat.

B. Kerugian

1. Memiliki data yang kompleks
2. Menggunakan perangkat lunak yang mahal.
3. Terjadi penurunan pada kecepatan akses data sehingga membutuhkan alokasi memori yang besar.

2.1.1.5. DB2

2.1.1.5.1. Produk DB2

DB2 merupakan DBMS pertama buatan IBM yang di buat pertama kali pada tahun 1983. DB2 mendukung penyimpanan berbagai macam tipe data, seperti *audio*, *video*, dan teks. Hingga pada versi DB2 *9.7*, dengan nama sandi Cobra, DB2 dapat beroprasi diberbagai macam *platform*, seperti AIX, HP-UK, LINUX, Solaris, Windows, i5/OS dan z/OS.

2.1.1.5.2. Konsep dan Teknologi DB2

Menurut Chong R.F., Wang X., Dang M., et.al (2008:4) IBM memiliki beberapa strategi yang digunakan pada DB2 berdasarkan konsep dan strategi yang telah ditentukan, yaitu :

1. *On-Demand Business*

   Dengan perkembangan teknologi yang sangat pesat pada saat ini, dimana

pelanggan yang semakin tergantung dan menginginkan semakin kecilnya kesalahan pada setiap sistem bisnis yang dijalankan, DB2 dirancang lebih terintegrasi dan fleksibel, dimana bisnis proses suatu perusahaan harus terintegrasi dengan perusahaan lainnya secara baik, baik antar *client*, mitra utama, *supplier*, dan pelanggan dengan respon yang lebih cepat dalam menanggapi kebutuhan pelanggan, peluang pasar, dan gangguan dari luar.

2. *Information On Demand*

   DB2 telah menyediakan berbagai macam informasi, *tools*, aplikasi yang dibutuhkan pelanggan.

3. *Service - Oriented Architecture*

   Service - Oriented Architecture (SOA) merupakan metodologi dalam merancang sistem agar menjadi lebih terintegrasi, fleksibel, dan mengurangi penggandaan komponen.

   Dengan menggunakan metodologi ini secara baik, aktifitas bisnis diperlakukan seperti suatu layanan yang dapat diakses

dan bergantung dengan suatu jaringan. Dalam penggunaannya SOA memiliki beberapa tahapan, yaitu :

a. *Model*

Pada tahapan ini merupakan tahapan pemodelan dan pengoptimalisasi suatu sistem bisnis. Tahapan ini digunakan untuk menentukan sistem bisnis yang dibutuhkan dan tipe data yang dapat diakses.

b. *Assemble*

Tahapan ini merupakan pembangunan sistem bisnis yang baru dan mendaur ulang suatu sistem yang lama untuk digabungkan menjadi suatu aplikasi baru.

c. *Deploy*

Pada tahapan ini merupakan tahapan dimana suatu aplikasi diberikan kepada orang-orang yang terhubung langsung dengan sistem bisnis.

d. *Manage*

Pada tahapan ini pengembang *software* harus mengelola dan memeriksa sistem

dengan memeriksa dan memperbaiki masalah terhadap ketidak efisienan suatu sistem bisnis yang dibuat.

e. *Governance*

Memastikan seluruh tahapan yang ada dilakukan dengan baik, baik dari sisi dalam atau luaran dari organisasi.

4. *Web Service*

Web Service merupakan layanan yang dapat digunakan di dalam *web*. *Web service* adalah aplikasi yang menggunakan jaringan. Melalui penggunaan *web service, user* tidak perlu memperhatikan lokasi penggunaan, dan bahasa yang digunakan. *Web service* lebih kuat karena informasi yang dibuat sangat minimal untuk dilakukan perubahan dan tidak ada campur tangan manusia dalam setiap proses penggunaannya.

2.1.1.5.3. DB2 Express-C

DB2 Express-C merupakan versi DB2 yang digunakan pada suatu komunitas atau perusahaan. Setiap aplikasi bisnis dapat menggunakannya secara

gratis dengan mengkoneksikan aplikasi yang di buat ke *data server* yang menggunakan DB2 express-C.

2.1.1.6. *Structured Query Language*

Structured Query Language (SQL) merupakan bahasa komputer standard ANSI (*American National Standard Institude*). Dengan SQL *user* dapat mengakses *database*, menjalankan *query* untuk mengambil data dari *database*, menambahkan data ke *database*, menghapus data di dalam *database*, dan ubah data di dalam *database*.

2.1.1.7. *Entity Relational Diagram*

Entity Relational Diagram (ERD) menurut Whitten, Bentley dan Dittman (2004:295) adalah model data yang memanfaatkan beberapa notasi untuk menggambarkan hubungan antar data.

ERD merupakan suatu pemodelan dari basis data relasional yang didasarkan atas persepsi di dalam dunia nyata, dunia ini senantiasa terdiri dari sekumpulan objek yang saling berhubungan antara satu dengan yang lainnya. Suatu objek disebut entitas dan hubungan yang dimilikinya disebut *relationship*. Suatu entitas bersifat unik dan memiliki atribut sebagai pembeda dengan entity lainnya.

Dalam ERD untuk memodelkan struktur data dan hubungan antar data digunakan beberapa notasi dan simbol. Terdapat tiga simbol yang digunakan, yaitu :

A. Entitas

Menurut Whitten, Bentley dan Dittman (2004:295) entitas atau *entity* diartikan sebagai kelompok orang, tempat, objek, kejadian atau konsep tentang apa yang di perlukan untuk menyimpan data.

Ada dua jenis macam entitas, yaitu:

- *Strong Entity Type* adalah entitas yang keberadaanya tidak tergantung pada entitas lain. Terkadang disebut *parent, owner dominant.*
- *Weak Entity Type* adalah entitas yang keberadaanya bergantung pada entitas lain. Disebut juga *child dependent, subordinate.*

B. Atribut

Atribut adalah ciri-ciri kualitatif yang dimiliki oleh suatu objek yang mencerminkan suatu objek tersebut.

Menurut Whitten, Bentley dan Dittman (2004:295), atribut diartikan sebagai sifat atau karakteristik deskriptif suatu entitas.

Dalam sebuah entitas harus memiliki atribut yang disebut *key*. *Key* adalah atribut atau kumpulan dari atribut yang mengasumsikan nilai unik untuk setiap entitas. Terdapat beberapa jenis-jenis *key*, yaitu :

1. *Candidate Key*

   Merupakan *key* yang akan menjadi *primary key* di dalam suatu entitas.

2. *Primary Key*

   Merupakan *key* yang akan menjadi salah satu identitas *key* yang unik untuk menggambarkan suatu entitas.

3. *Alternate Key*

   Merupakan *Candidate key* yang tidak terpilih untuk menjadi *Primary key*.

4. Foreign Key

Primary key suatu entitas yang digunakan dalam entitas lain untuk mengidentifikasi contoh hubungan.

C. Relasi atau hubungan

Menurut Whitten, Bentley dan Dittman (2004:298) Relasi ialah hubungan bisnis yang ada diantara satu atau lebih entitas.

Hubungan tersebut dapat merepresentasikan kejadian yang menghubungkan entitas atau hanya persamaan logika yang terdapat diantara entitas.

Setiap relasi antar entitas memiliki sebuah derajat relasi atau kardinalitas. Kardinalitas ialah jumlah minimum dan maksimum kemunculan suatu entitas yang dihubungkan dengan kemunculan tunggal entitas lain. Jenis-jenis kardinalitas antara lain :

- Satu ke Satu (*One to One*)

  Setiap anggota entitas A, hanya dapat berhubungan dengan satu entitas B tetapi tidak sebaliknya.

- Satu ke Banyak (*One to Many*)

  Setiap anggota entitas A dapat berhubungan dengan lebih dari satu anggota entitas B tetapi tidak sebaliknya.

- Banyak ke banyak (*Many to Many*)

  Setiap entitas A dapat berhubungan dengan banyak entitas, himpunan entitas B dan demikian juga dengan sebaliknya.

2.1.2. *Internet*

2.1.2.1. Pengertian *Internet*

Menurut Wira (2009). *Internet* berasal dari kata *interconnected-networking* yang merupakan jaringan global yang dapat menghubungkan suatu jaringan dengan jaringan lainnya menggunakan beberapa media seperti kabel, kanal satelit maupun frekuensi radio.

Jaringan *internet* bekerja berdasarkan suatu *protocol* atau aturan yang telah ditetapkan. TCP/IP (*Transmission Control Protocol Internet Protocol)* adalah *protocol* yang umum digunakan agar dapat menghubungkan suatu komputer

dengan komputer lainnya di mana setiap komputer diberikan nomor unik yang disebut nomor IP.

2.1.2.2. Sejarah *Internet*

Internet memiliki sejarah yang cukup panjang dari awal ditemukannya sampai saat ini. Sejak ditemukan sekitar tahun 1969, *internet* mengalami banyak pengembangan dan perbaikan hingga akhirnya bisa seperti saat ini.

Internet merupakan jaringan komputer yang dibentuk oleh Departemen Pertahanan Amerika Serikat pada tahun 1969, melalui proyek ARPA yang disebut ARPANET (*Advanced Research Project Agency Network*), di mana pemerintah Amerika Serikat mendemonstrasikannya dengan *hardware* dan *software* komputer yang berbasis UNIX, *internet* dapat melakukan komunikasi dalam jarak yang tidak terhingga melalui saluran telepon. Proyek ARPANET merancang bentuk jaringan, kehandalan, seberapa besar informasi dapat dipindahkan, dan akhirnya semua standar yang mereka tentukan menjadi cikal bakal pembangunan protokol baru yang sekarang dikenal sebagai TCP/IP *(Transmission Control Protocol / Internet Protocol).*

2.1.3. HTTP

Hypertext Transfer Protocol (HTTP), merupakan suatu protokol yang digunakan oleh *World Wide Web*. HTTP mendefinisikan bagaimana suatu pesan bisa diformat dan dikirimkan dari *server* ke *client*.

2.1.4. *Uniform Resource Language*

URL (*Uniform Resource Locator*) adalah rangkaian karakter menurut suatu format standar tertentu, yang digunakan untuk menunjukkan alamat suatu sumber seperti dokumen dan gambar di*internet*.

URL pertama kali diciptakan oleh Tim Berners-Lee pada tahun 1991 agar para penulis dokumen dapat mereferensikan pranala ke *World Wide Web*.Sejak 1994, konsep URL telah dikembangkan menjadi istilah *Uniform Resource Identifier* (URI) yang memiliki sifat lebih umum dalam penggunaannya.

2.1.5. *World Wide Web*

Menurut Kristianto (2002), *World wide web* adalah semua bagian *internet* yang dapat diakses dengan *software web browser*. *World wide web* (www) terdiri dari jutaan situs *web* dan dari *web* tersebut terdiri dari halaman-halaman *web*. Salah satu keunggulan www adalah *hyperlink*. *Hyperlink* ialah teks yang pada umumunya berwarna biru atau bergaris bawah yang dapat diklik dan akan menuju ke halaman *web* yang lain. *Hyperlink* ini dapat diletakkan di manapun

di dalam halaman *web* dan boleh diatur untuk menuju ke mana saja diseluruh *web*.

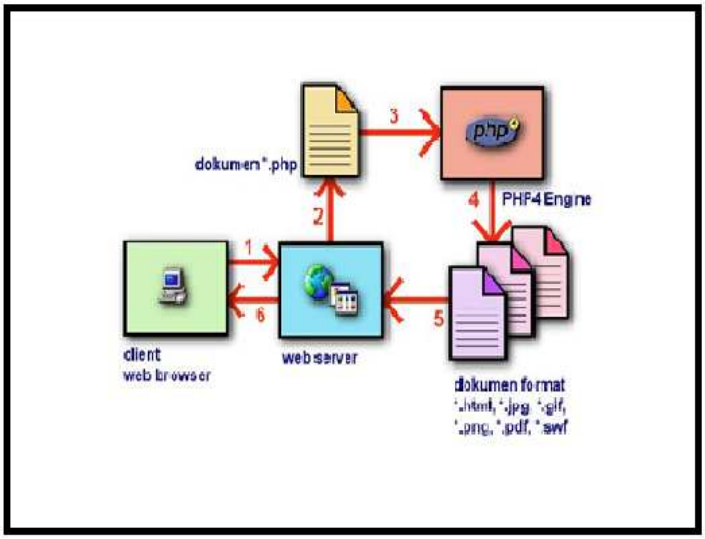
2.1.5.1. *Web Server*

Web server merupakan program aplikasi yang memiliki fungsi sebagai tempat penyimpanan dokumen yang terdapat didalam *web*. Dokumen *web* yang ditulis menggunakan metode *client-side scripting* atau *server-side scripting* disimpan di direktori utama *web server* (*document root*). Contoh aplikasi *web server* yang umum digunakan saat ini ialah *Apache, Web server apache tomcat, Web server microsoft internet information.*

2.1.5.2. *Server Side Scripting*

Server side scripting adalah bahasa pemrograman *web* yang pengolahannya dilakukan disisi *server*. Pada metode ini *server* adalah *web server* yang di dalamnya telah mengintegrasikan komponen *web engine*. *Server side scripting* adalah HTML *embedded,* yaitu semua *server side scripting* dapat disisipkan ke dalam dokumen *web* yang menggunakan HTML atau sebaliknya.

Contoh *server side scripting*:*Active Server Pages (ASP), PHP : Hypertext Preprocessor (PHP), Java Server Pages (JSP).*



Gambar 2.1 Proses *Server Side Scripting*

Sumber: http://ikc.dinus.ac.id/berseri/ivan-php/php-mudah01.php

2.1.5.3. *Browser Web*

Browser merupakan *software* atau alat yang digunakan untuk menjelajahi *internet*. Pengertian browser sejalan dengan adanya istilah "browse". Menurut *Cambridge Dictionary online"browse"* adalah"*to look at or through something to see what is there*" yang di terjemahkan ke dalam bahasa Indonesia adalah"Untuk melihat atau mencari sesuatu untuk melihat apa yang ada". Arti *browser* oleh beberapa kalangan disamakan pula sebagai "perambah".

Menurut Arief (2011:19), *web browser* merupakan program yang berfungsi untuk menampilkan dokumen-dokumen *web* dalam format HTML. *Web browser* dibuat

menggunakan standarisasi yang dibuat oleh *World Wide Web Consortium* (W3C) yang merupakan badan tunggal yang mengurus semua hal yang berkaitan dengan *web*. Berikut Contoh *Web browser* yang cukup populer beserta *web engine* yang di gunakan :

1. *Web Engine WebKit* : Safari, Google Chrome
2. *Web Engine Trident* : Microsoft *Internet* Exploler
3. *Web Engine Gecko* : Mozila Firefox
4. *Web Engine Presto* : Opera

2.1.6. Sistem Informasi

2.1.6.1. Pengertian Sistem

Sistem adalah suatu kesatuan prosedur atau komponen yang saling berkaitan satu dengan yang lainnya bekerja bersama sama sesuai dengan aturan yang diterapkan sehingga membentuk suatu tujuan yang sama, dimana dalam sebuah sistem bila terjadi satu bagian saja yang tidak bekerja atau rusak maka suatu tujuan bisa terjadi kesalahan hasilnya atau outputnya.

Sistem menurut Stair dan Reynolds (2010:8) ialah suatu set elemen atau komponen yang saling berinteraksi untuk mencapai suatu tujuan. Elemen didalam sistem saling bekerja sama tergantung dari bagaimana sistem kerja di buat. Sistem memiliki masukan, proses, mekasnisme, hasil dan timbal balik dari proses yang telah di jalankan.

2.1.6.2. Pengertian Informasi

Informasi berasal dari bahasa Perancis kuno "*informacion"* pada tahun 1387 yang diambil dari bahasa Latin "*informationem"* yang berarti "garis besar, konsep, ide". Informasi merupakan kata benda dari informare yang berarti aktivitas dalam "pengetahuan yang di komunikasikan".

Informasi dapat di artikan sebagai data yang telah di olah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya.

Menurut Stair dan Reynolds (2010:5), Informasi merupakan kumpulan data fakta yang terorganisasi sehingga dapat meberikan nilai tambah terhadap suatu data.

2.1.6.3. Pengertian Sistem Informasi

Sistem informasi menurut Stair dan Reynods (2010:4) ialah merupakan suatu kumpulan elemen yang saling berhubungan yang dapat di ambil, manipulasi, simpan dan menyebarluaskan data untuk mendapatkan timbal balik dari sistem yang di buat.

2.1.6.4. Siklus Pengembangan Sistem

Menurut Stair dan Reynolds (2010:496) Siklus pengembangan suatu sistem atau yang lebih di kenal dengan metode *System Development Life Cycle* (SDLC), terdiri dari beberapa tahapan yaitu:

1. Investigasi

   Merupakan fase pengembangan sistem di mana dilakukan identifikasi masalah dan kesempatan untuk menyelesaikan suatu masalah.

2. Analisis

   Merupakan fase pengembangan sistem yang menentukan informasi yang harus diselesaikan dengan mempelajari sistem yang telah ada dan cara kerja untuk mengidentifikasi kekuatan, kelemaham, dan kesempatan untuk memperbaiki suatu sistem.

3. Desain

   Merupakan fase di dalam pengembangan sistem yang mendefinisikan bagaimana suatu sistem informasi akan digunakan sesuai dengan masalah yang ada.

2.2. Teori Khusus

2.2.1. Sistem Billing

2.2.1.1. Definisi Sistem Billing

Menurut dictionary.cambridge.org , *Bill* adalah *a request for payment of money owed, or the piece of paper on which it is written (noun); to give or send someone a bill asking for money that they owe for a product or service (verb).* Dalam bahasa Indonesia, *Bill* berarti permintaan untuk pembayaran yang terhutang, atau selembar kertas yang telah yang ditulis (kata benda); untuk memberi atau mengirim seseorang sebuah

permintaan tagihan untuk membayar sejumlah uang yang diutangkan untuk sebuah produk atau layanan (kata kerja).

Sistem billing merupakan suatu sistem yang mengatur seluruh penghitungan biaya yang berkaitan dengan suatu layanan jasa. Menurut Mostafa (2005) sistem billing memproses seluruh kegiatan telekomunikasi yang digunakan dan akan menjadi suatu laporan berbentuk *Call Detailed Record* (CDR). Proses billing menerima berbagai catatan billing dari berbagai proses telekomunikasi dimana sistem harus menentukan tingkat penagihan yang terkait dengan catatan billing, penghitungan biaya setiap *record* penagihan, menggabungkan catatan-catatan berkala untuk menghasilkan faktur, mengirim faktur kepada pelanggan, serta menerima uang yang diterima dari pelanggan.

2.2.1.2. Mekanisme Sistem Billing

Menurut Garko A.B. dan Wajiga G.M. (2012) terdapat dua mekanisme yang digunakan pada sistem billing, yaitu:

1. *Post-paid*

   Memiliki mekanisme bahwa pelanggan membayar secara langsung layanan yang mereka gunakan. Dalam penggunaan layanan *post-paid* pihak penyelenggara telekomunikasi harus efisien dan akurat dalam penentuan tarif didalam sistem billing.

2. *Pre-paid*

Dalam penggunaannya *user* harus melakukan pembelian credit atau voucher terlebih dahulu. Setelah pembelian dilakukan *user* dapat menggunakan layanan sesuai dengan ketentuan dan apabila kredit mereka sudah habis maka penggunaan layanan akan dibatasi.

2.2.1.3. Manfaat Sistem Billing

Manfaat menggunakan sistem billing adalah sebagai berikut :

1. *Set up Billing Codes*

Nomor *bill* atau kuitansi dapat dihasilkan secara otomatis oleh sistem (*autogenerate*) sehingga mengurangi tingkat kecurangan.

2. *Set up recuirring Charge Schedule*

Dengan adanya sistem billing *user* dapat melakukan perubahan pada jadwal penagihan iuran seperti penagihan setiap bulan, setiap caturwulan, atau setiap semester.

3. *Late-fee Assesment*

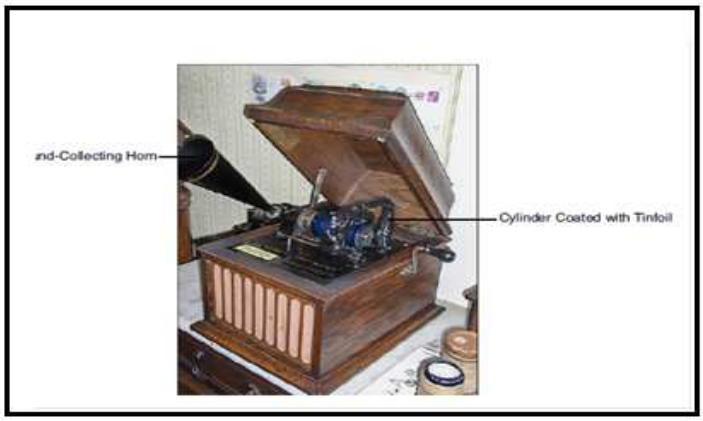
Metode ini digunakan untuk mengetahui dengan cepat *user* yang belum membayar sehingga pengawasan dapat dilakukan dengan efektif dan efisien.

4. *Statement*

Sistem Billing akan mencetak beberapa jenis billing statements yang berisi rekapitulasi mengenai pembayaran yang bersifat historis. Dengan adanya metode ini pihak penyelenggara telekomunikasi dapat mengurus tagihan sesuai dengan tahun kalender.

2.2.2. Teknologi Telepon

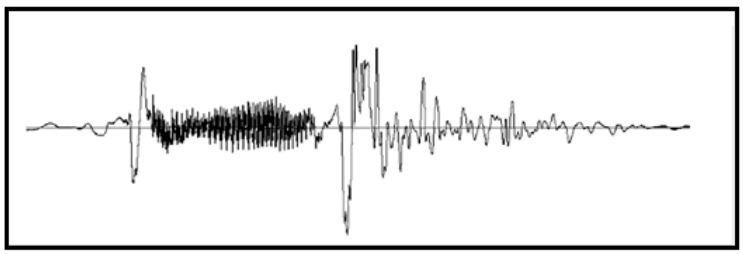
Telepon berkembang dimulai pada tahun 1877 yang di temukan oleh Thomas Edison. Pada awal kemunculannya telepon masih disebut *phonograph*, alat ini dapat merekam suara dengan cara menggunakan jarum yang di tekan pada suatu piringan yang terbuat dari kertas timah yang akan menyusun seluruh suara percakapan yang ada. *Phonograph* juga dapat mengulang kembali suara yang telah terekam dengan memindahkan titik jarum yang diinginkan dengan menggunakan kecepatan putaran konstan. Bentuk rekaman ini merupakan representasi dari sistem sinyal *analog*.



Gambar 2.2 Replika *Phonograph*

Sumber : Cioara,Valentine (2012 : 6)

Sinyal *analog* di gunakan untuk menangkap sinyal *audio* untuk menyampaikan informasi *audio* yang dikirim. Dalam era saat ini di mana pada umumnya teknologi terhubung melalui beberapa bentuk kabel, arus listrik yang di gunakan untuk mengirim sinyal *analog*. Ketika berbicara menggunakan telepon *analog*, suara yang di keluarkan di ubah menjadi arus listrik. *Volume* dan *pitch* yang di keluarkan akan menghasilkan beberapa variasi arus listrik.

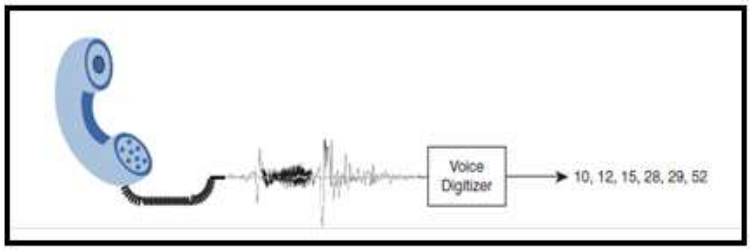


Gambar 2.3 Contoh hasil Variasi Suara

Sumber : Ciorra, Valentine (2012 : 7)

Sinyal *analog* memiliki beberapa kekurangan, yaitu sinyal *analog* sering mengalami degradasi (pemudaran sinyal) jarak jauh. Untuk meingkatkan jarak sinyal harus digunakan repeater yang di gunakan untuk menguatkan sinyal *analog*. Namun *reapeater* memiliki beberapa kekurangan yaitu perangkat ini tidak dapat membedakan antara suara yang dikirim atas kawat dan garis kebisingan, sehingga semakin banyak pengulangan dalam penggunaan repeater maka semakin sulit untuk memahami sinyal yang dikirim. Selain itu sinyal *analog* juga sulit dalam mendukung jaringan dengan geografis yang luas. Karena setiap antar telepon dibutuhkan dua kawat, dan bundel.

Untuk mengatasi masalah tersebut maka di ciptakan sinyal *digital*. Sinyal *digital* menggunakan angka untuk mewakili tingkat suara. Pada dasarnya setiap angka yang digunakan pada sinyal *digital* merupakan suara yang dihasilkan selama penelpon berbicara. Sinyal *digital* menggunakan teknologi *Time-division multiplexing* (TDM). TDM memungkinkan suatu jaringan untuk dapat membawa beberapa percakapan pada waktu bersamaan melalui empat kawat, satu jalan. Proses tersebut dapat dilakukan karena percakapan telah diubah kedalam nilai-nilai numerik yang telah didigitalkan dan ditransmisikan kedalam beberapa slot tertentu.



Gambar 2.4

Contoh Perubahan sinyal *analog* menjadi *digital*

Sumber : Ciorra, Valentine (2012 : 10)

2.2.2.1. *Cisco Unified Communication Manager*

Menurut Broklund L. dan Bailey T. (2011), *Cisco Unified Communication Manager* (CUCM) merupakan teknologi komunikasi berbasis *IP Telephony* yang dapat mengirimkan informasi secara efektif dan efisien. Teknologi ini menyediakan berbagai macam layanan seperti, telepon, *e-mail*, *voicemail*, *web-confrance*, *instant messaging*, *mobile phone*, dsb.

Menurut jurnal *Cisco System* (2011), *Cisco Unified Communication manager* (CUCM) merupakan suatu perangkat telekomunikasi yang memungkinkan data, suara, *video* dapat ditransmisikan ke suatu jaringan dengan menggunakan IP yang telah ditentukan. *Cisco Unified Communication manager* dirancang untuk memaksimalkan proses komunikasi dengan mengurangi konfigurasi,

pemeliharaan yang dibutuhkan dan menyediakan interoperabilitas dengan berbagai jenis aplikasi.

2.2.2.2. *IP-Telephony*

IP-Telephony adalah suatu aplikasi teknologi yang mentransmisikan komunikasi suara melalui jaringan yang menggunakan suatu IP yang telah ditentukan. Hubungan *IP-Telephony* dengan *Cisco Unified Communication Manager* ialah *Cisco Unified Communication Manager* merupakan produk *software* dan *hardware*, sedangkan *IP-Telephony* merupakan aplikasi seperti *voice-messaging system*, *video device,* dan aplikasi lainnya.

2.2.2.3. VoIP

VoIP adalah teknologi yang memanfaatkan *Internet Protocol* untuk menyediakan komunikasi suara secara elektronis dan *real-time*. Suara yang merupakan data *analog* diubah menjadi *data digital* oleh *decoder*, *data digital* tersebut di-*compress* dan di-*transmit* melalui jaringan IP. VoIP menggunakan IP sebagai “basic transport”. Pada layer Transport, VoIP menggunakan TCP dan UPD over IP. Karena data dikirimkan melalui protokol IP, maka data dikirimkan secara ‘*Switcing Packet*’ yaitu data dipecah menjadi paket-paket. Informasi dibagi-bagi dalam paket yang panjangnya tertentu, kemudian tiap paket dikirimkan secara individual.

2.2.3. *Call Detailed Record*

2.2.3.1. Pengertian *Call Detailed Record*

Menurut jurnal Cisco system (2007) *Call Detailed Record* (CDR) merupakan suatu laporan hasil dari proses suatu komunikasi yang mendefinisikan nomor panggilan, tempat panggilan, tanggal dan waktu di mulainya panggilan, dan waktu kapan panggilan tersebut dimulai dan d iakhiri, pada CDR telah menetapkan *node-node* yang menjelaskan seluruh sistem parameter yang terjadi setiap proses telekomunikasi terjadi.

2.2.3.2. Karakteristik *Call Detailed Record*

Menurut Breda G. , Mendes S. ,Bottoli L. (2007), CDR memiliki beberapa karakteristik, diantaranya :

1. CDR memiliki data yang sangat banyak yang berhubungan langsung kedalam beberapa jenis laporan data.
2. CDR memiliki data yang bersifat historis yaitu menyimpan seluruh data yang terjadi pada proses telekomunikasi baik yang sedang terjadi atau telah terjadi.
3. CDR bersifat *reliability* yaitu dapat memungkinkan pihak penyelenggara layanan telekomunikasi untuk menggunakan laporan dan mendapatkan hasil analisis data yang lebih baik.

2.2.4. Teknologi yang digunakan

2.2.4.1. XAMPP

XAMPP merupakan *tool* yang menyediakan paket perangkat lunak ke dalam satu buah paket. Dalam paket tersebut sudah terdapat *apache (web server), MySQL (Database), PHP (Server Side), Perl, FTP sever*, dan berbagai *software* lainnya.

2.2.4.2. Perl

Perl merupkan bahasa pemograman yang di temukan pada tahun 1987 oleh Larry Wall. Menurut Pearce (2002:3) *perl* di gunakan untuk mengoptimalisasikan bahasa di dalam membaca *arbitrary text file*, mengekstrak informasi dari berbagai teks file, dan membuat laporan berdasarkan laporan yang ada. Perl merupakan kombinasi bahasa yang sudah sangat familiar bagi para pengguna seperti *C, sed, awk,* dan *sh*.

2.2.4.2.1 Strawberry Perl

Strawberry perl merupakan bahasa pemograman *perl* yang digunakan untuk *windows platform*. Apabila pemograman *perl* lainnya tergantung dengan pengembangan perangkat lunak yang telah diatur untuk menginstall, Strawberry *perl* telah terkonfigurasi dan dikemas dengan baik.

Strawberry perl dapat di gunakan dengan mengunduh beberapa modul yang telah disediakan, namun terdapat beberapa modul seperti *XS modul* harus membutuhkan *C compiler* untuk mendukung proses instalasi dapat diselesaikan.

2.2.4.3. PHP

PHP yang merupakan *Hypertext processor* merupakan bahasa skrip yang terintegrasi dengan HTML yang bersifat *open source* dan *server-side* yang sering digunakan untuk menciptakan *web* dinamis. PHP merupakan bahasa yang berbentuk skrip yang ditempatkan dalam *server* dan diproses didalam *server* dan dikirim kepada sebuah klien untuk diimplementasikan menggunakan *browser*.

Kode PHP di simpan sebagai *plain text* dalam format *ASCII*, sehingga kode PHP dapat ditulis disemua *editor text* seperti *windows notepad, windows wordpad,* dsb.

Menurut Gilmore (2008:2) PHP muncul pada tahun 1995 ketika seorang *software development* bernama Rasmus Lerdorf mengembangkan bahasa *Perl/CGI* yang digunakan untuk mengetahui seberapa banyak orang yang melihat daftar riwayat hidupnya. Skrip-skrip inilah yang menjadi acuan dalam pembangunan PHP. Lerdof melakukan perubahan awal dengan menciptakan PHP/FI versi 2. Pada versi ini pemograman dapat

menempelkan kode terstruktur kedalam *tag HTML*. Hingga saat itu PHP menjadi bahasa yang sangat popular. Dalam pengembangannya selanjutnya, dengan mempertahankan konsep awal digabungkan langsung bersama HTML dengan menggunakan parsing sehingga terciptalah PHP versi 3 dimana terdapat 50.000 orang lebih menggunakan PHP pada saat itu untuk dijadikan *web*-nya. Selanjutnya pengembangan dilanjutkan dengan meningkatkan performa kecepatan dari PHP agar dapat digunakan dalam jangka waktu yang panjang.

2.2.4.4. *Hyper Text Markup Language*

HTML (*Hyper Text Markup Language*) adalah sebuah *bahasa markup* yang digunakan untuk membuat sebuah halaman *web* dan menampilkan berbagai informasi di dalam sebuah *browser internet.*

HTML merupakan perkembangan dari bahasa SGML (*Standart Generalized Markup Language*). HTML merupakan bahasa standar yang digunakan secara luas untuk menampilkan halaman *web*.

2.2.4.5. *Cascading Style Sheet*

Cascading Style Sheets (CSS) adalah suatu bahasa *stylesheet* yang digunakan untuk mengatur tampilan suatu dokumen yang ditulis dalam bahasa *markup*.Penggunaan yang paling umum dari CSS adalah untuk memformat halaman *web* yang ditulis dengan HTML dan XHTML.

CSS digunakan untuk menentukan warna, jenis huruf, tata letak, dan berbagai aspek tampilan dokumen. CSS digunakan terutama untuk memisahkan antara isi dokumen (yang ditulis dengan HTML atau bahasa *markup* lainnya) dengan presentasi dokumen (yang ditulis dengan CSS). Pemisahan ini dapat meningkatkan aksesibilitas isi, memberikan lebih banyak keleluasaan dan kontrol terhadap tampilan, dan mengurangi kompleksitas serta pengulangan pada stuktur isi.

CSS memungkinkan halaman yang sama untuk ditampilkan dengan cara yang berbeda untuk metode presentasi yang berbeda, seperti melalui layar, cetak, suara (sewaktu dibacakan oleh *browser* basis-suara atau pembaca layar), dan juga alat pembaca braille. Halaman HTML atau XML yang sama juga dapat ditampilkan secara berbeda, baik dari segi gaya tampilan atau skema warna dengan menggunakan CSS.

Menurut Septian (2011) *Cascading Style Sheets (CSS)* merupakan bahasa pemograman *stylesheet* yang digunakan untuk mengatur tampilan suatu dokumen dalam bentuk *web* atau aplikasi dalam bahasa *markup* yang biasanya ditulis dengan format HTML atau XHTML.

2.2.4.6. *Secure Shell*

2.2.4.6.1. Definisi *Secure Shell*

Menurut Krishna, Jamwal, Chaitanya, et al. (2011) *Secure Shell* (SSH) adalah protokol yang menfasilitasi komunikasi antara dua sistem yang menggunakan sistem arsitektur *client / server* dan memungkinkan pengguna sistem untuk *login* ke sistem *host server* jarak jauh. Tidak seperti protokol komunikasi lainnya seperti FTP atau telnet, SSH dapat mengenkripsi sesi *login* sehingga dapat menghindari penyusup atau *hacker* untuk mengambil *password* yang terenkripsi.

SSH dirancang untuk menggantikan sistem protokol sebelumnya yang memiliki aplikasi terminal kurang aman digunakan untuk *login* ke *host remote*, seperti telnet atau rsh.

2.2.4.6.2. Fitur-fitur *Secure Shell*

Protokol SSH telah menyediakan pengaman sebagai berikut :

- Setelah koneksi awal, klien dapat menverifikasi bahwa sistem terhubung ke server yang sama yang telah terhubung sebelumnya.
- Klien mengirimkan informasi autentifikasi kepada *server* menggunakan proses yang cepat, 128-bit enkripsi.
- Semua data dikirim dan diterima selama session dikirim menggunakan 128-bit enkripsi yang menyebabkan pengiriman sulit dihalangi untuk dideskripsikan dan dibaca.
- Klien dapat meneruskan XII [2] aplikasi dari *server*. Teknik ini disebut *XII forwading*, yaitu menyediakan sarana yang aman untuk menggunakan aplikasi grafis melalui jaringan. Karena SSH bekerja untuk mengengkripsi semua hal yang dikirim dan diterima, maka SSH dapat

digunakan utuk mengamankan sistem walaupun protokol yang tidak aman.

2.2.4.7. *Adobe Dreamweaver*

Adobe Dreamweaver menjadi *web editor* yang banyak di gunakan oleh para *web developer*. Hal itu antara lain karena kemudahan dalam penggunaannya, kelengkapan fitur dan juga dukungannya terhadap teknologi terkini. *Dreamweaver* merupakan salah satu perangkat lunak yang dikembangkan oleh *Macromedia dreamweaver* merupakan perangkat lunak yang ditujukan untuk membuat suatu situs *web*. Versi pertama di rilis pada tahun 1997, dan sejak itu *Inc* , hingga sekarang resmi di milik *Adobe*.

2.2.5. *Unified Modeling Language*

2.2.5.1. Definisi *Unified Modeling Language*

Menurut Booch G., Maksimchuk R. A., Engle M.W., et. al (2007:147), UML (*Unified Modeling Language*) merupakan bahasa standar yang digunakan untuk mendefinisikan, menspesifikasi, membangun dan mendokumentasikan objek dari suatu sistem piranti lunak, baik dalam suatu perancangan *Software* bisnis atau *non-Software*. UML menyediakan sekumpulan tata cara terbaik dalam proses perancangan suatu sistem yang luas dan kompleks. UML memiliki notasi yang terdefinisi dengan baik

sehingga dapat memungkinkan bagi perancang sistem peranti lunak untuk menggambarkan skenario atau merumuskan arsitektur sistem sehingga dapat dikomunikasikan dengan baik kepada orang lain dalam mengambil keputusan.

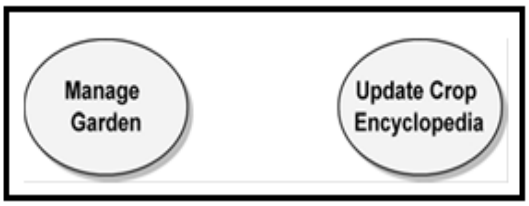
2.2.5.2. *Use Case Diagram*

Menurut Menurut Booch G., Maksimchuk R. A., Engle M.W., et. al (2007:175) *Use Case Diagrams* merupakan diagram yang menggambarkan konteks suatu sistem yang akan dibangun dan fungsi yang disediakan oleh sistem. *Use case diagrams* memperlihatkan hubungan antara aktor (*Actors*) dengan sistem (*Case*).

Gambar 2.5 Contoh *Actor*

Booch G., Maksimchuk R. A., Engle M.W., et. al. (2007:176)

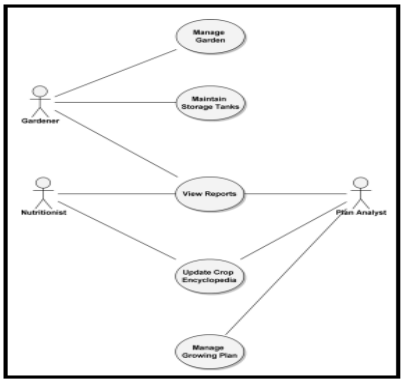
Aktor menggambarkan *user* yang akan berinteraksi dengan sistem yang dibuat.



Gambar 2.6 Contoh *Case*

Booch G., Maksimchuk R. A., Engle M.W., et. al (2007:177)

Use case adalah beberapa aksi atau tugas yang dilakukan *user* yang dapat dilakukan untuk menyelesaikan suatu masalah.



Gambar 2.7

Contoh *Use Case Diagrams*

Booch G., Maksimchuk R. A., Engle M.W., et. al

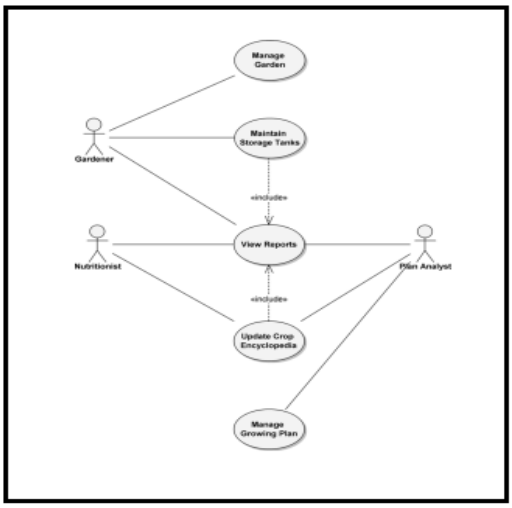
(2007:178)

2.2.5.2.1. *Include* dan *Enxtend Relation*

Diagram use case dapat menggunakan include dan extend yang digunakan untuk menjadikan diagram use case lebih baik dan mencegah penyalah gunaan antara aktor dan case.

1. << *include* >>

   Hubungan Include digunakan untuk menggambarkan kebutuhan suatu use case terhadap use case lainnya. Penggunaan include bertujuan untuk mengurangi pengulangan use case.



Gambar 2.8 Contoh penggunaan include

Booch G., Maksimchuk R. A.,

Engle M.W., et. al

(2007:181)

2. << extend >>

Extend digunakan untuk menggambarkan suatu use case merupakan perluasan dari use case lainnya. Extend digunakan jika sebuah use case (base use case) memiliki prilaku use case lain (extending use case).

Gambar 2.9 Contoh penggunaan extends

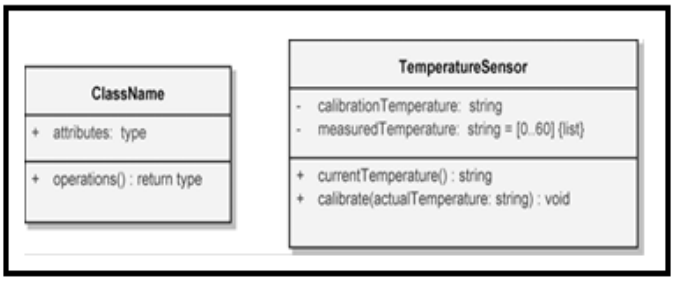
Booch G., Maksimchuk R. A.,

Engle M.W., et. al

(2007:182)

2.2.5.3. *Class Diagram*

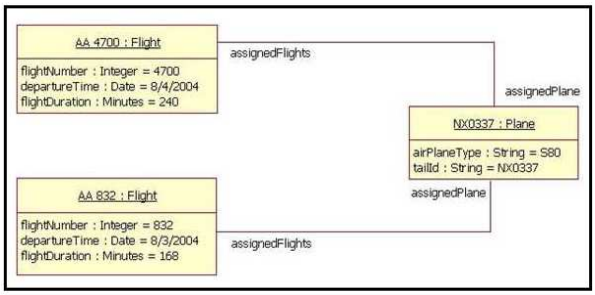
Class diagram merupakan diagram yang digunakan untuk mendeskripsikan tipe objek dalam suatu sistem dan hubungan kelas-kelas yang terdapat didalam suatu sistem secara logis. *Class diagram* terdiri dari tiga bagian penting yaitu kelas, atribut, dan operasi.



Gambar 2.10 Contoh class

Booch G., Maksimchuk R. A.,

Engle M.W., et. al (2007:192)



Gambar 2.11 Contoh Class Diagram

Sumber : (Bell,2004)

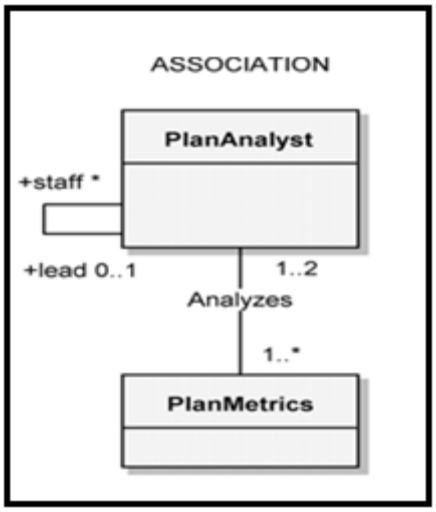
2.2.5.3.1. Hubungan Antar *Class*

Class di dalam sebuah *class* diagram dapat digabungkan dengan *class* lainnya. Metode penggambungan class dapat dilakukan dengan metode *Association, Generalization, Aggregation,* dan *Compotition*.

1. *Association*

   Association merupakan suatu metode penghubungan *class diagram* dengan menunjukan hubungan antar dua *class* serta sifat dari hubungan tersebut. *association* dapat diimplementasikan dengan menggunakan beberapa sintax yaitu :

   - 1 Hanya Satu
   - * Tidak terhingga (nol hingga tak terhingga)
   - 0..* Nol atau banyak
   - 1..* Satu atau banyak
   - 0..1 Nol atau satu
   - 3..7 Jarak yang lebih spesifik sesuai kebutuhan



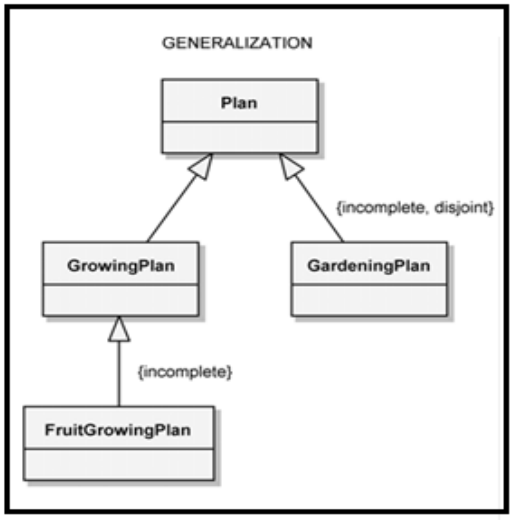
Gambar 2.12 Contoh Hubungan *Association*

Booch G., Maksimchuk R. A.,

Engle M.W., et. al (2007:195)

2. *Generalization*

Generalization menunjukan sebuah generalisasi dari hubungan *class* dimana pada metode ini digambarkan dengan panah tertutup. Panah mengarah ke *superclass,* sebaliknya *class* yang menurunkan *superclass* ialah *subclass*. *Subclass* mewarisi struktur dan prilaku *superclass* di mana setiap *subclass* memiliki satu atau banyak sifat warisan dari *superclass.*

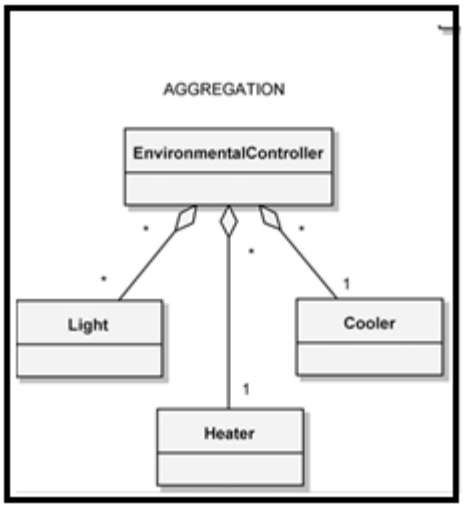


Gambar 2.13 Contoh Hubungan Generalization

Booch G., Maksimchuk R. A., Engle M.W., et. al (2007:195)

3. *Aggregation*

Hubungan *aggregation* digambarkan dengan mengunakan lambang *diamond* yang tidak terisi. *Diamond* akan menunjukan bahwa yang menjadi titik akhir ialah agregat sedangkan disisi lain ialah bagian dari agregat itu sendiri.



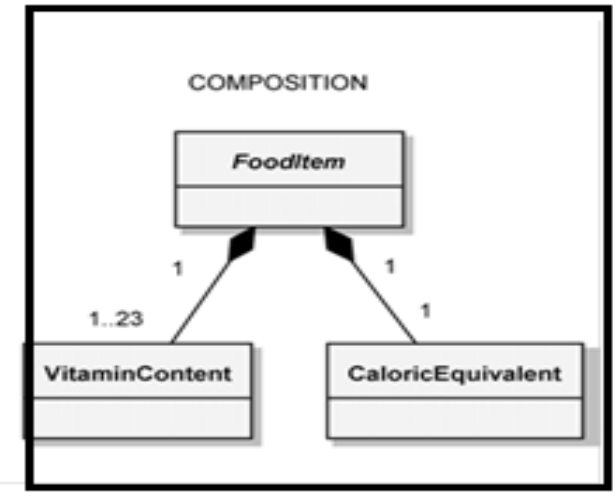
Gambar 2.14 Contoh Hubungan *Aggregation*

Booch G., Maksimchuk R. A.,

Engle M.W., et. al (2007:195)

4. *Composition*

Hubungan *composition* digunakan apabila diinginkan hasil yang bersifat lebih detail. *Composition* digambarkan dengan *diamond* yang terisi, di mana titik akhir tujuan menggambarkan agregat.



Gambar 2.15 Contoh Hubungan Compotition

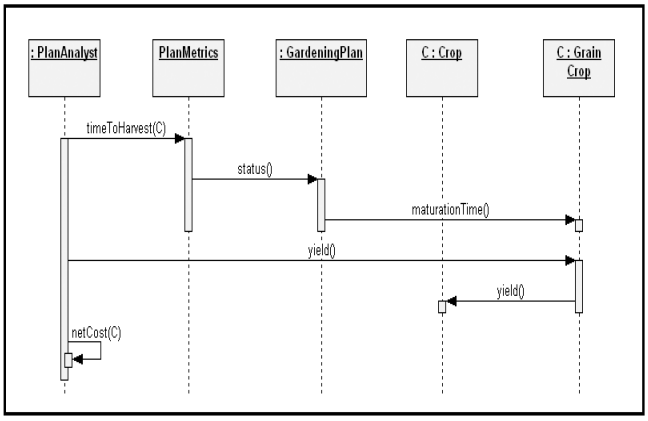
Booch G., Maksimchuk R. A.,

Engle M.W., et. al (2007:195)

2.2.5.4. *Sequence Diagram*

Menurut Menurut Booch G., Maksimchuk R. A., Engle M.W., et. al (2007:206), *Sequence Diagram* adalah diagram yang digunakan untuk menggambarkan urutan skenario dalam suatu sistem. Keuntungan menggunakan *sequence diagram* ialah mudah untuk dipahami dan menerjemahkan setiap skenario yang dibuat. *Sequence diagram* berfokus terhadap peristiwa yang bertentangan dengan sistem operasi, karena setiap peristiwa membantu untuk menentukan batas-batas dari sistem yang sedang dikembangkan.

Sequence diagram terbentuk dari *object* yang direpresentasikan dengan bentuk kotak yang berisi nama yang telah digaris bawahi, *message,* dan *time.*



Gambar 2.16 Contoh *Sequence Diagram*

Sumber: Menurut Booch G., Maksimchuk R. A., Engle M.W., et. al (2007:207)

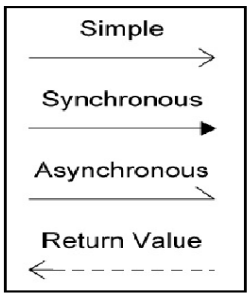
2.2.5.4.1. *Object*

Object pada *sequence diagram* merupakan entitas dari *sequence diagram* .*Object* digambarkan secara horizontal dibagian atas diagram. Terdapat garis vertikal putus-putus yang disebut *lifeline. Lifeline* selalu terdapat pada setiap *object* untuk mengindikasikan keberadaan *object.*

2.2.5.4.2. *Message*

Message ialah sebuah pesan yang disampaikan dari suatu *object* ke *object* lainnya melalui lifeline masing-masing *object*. *Message* dibagi dalam empat jenis, yaitu *Simple message, Synchronous message, asynchronous message,* dan *return value*. *Simple message* merupakan transfer

kontrol yang dikirim oleh suatu *object* ke *object* lain. *Simple message* digambarkan dengan panah yang memiliki dua garis pada kepala panahnya. *Synchronous message* merupakan transfer kontrol yang dikirim suatu *object* ke *object* lainnya dan akan menunggu jawaban dari *object* target untuk bisa melanjutkan ke proses selanjutnya. *Synchronous message* digambarkan dengan panah yang memiliki bentuk penuh pada kepalanya. *Asynchronous message* merupakan transfer kontrol yang dikirim oleh suatu *object* ke *object* lainnya dan tidak butuh menunggu untuk melanjutkan ke proses selanjutnya. *asynchronous message* digambarkan dengan bentuk panah yang memiliki bentuk setengah pada kepalanya. Sedangkan *return value* merupakan pesan pengembalian nilai dari suatu *object* di dalam *sequence diagram.*



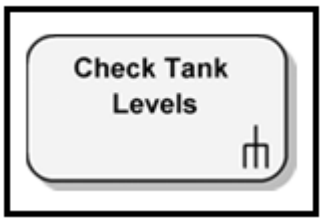
Gambar 2.17

Penggambaran *Message* pada *Sequence Diagram*

2.2.5.5. *Activity Diagram*

Menurut Booch (2007:185) *Activity Diagram* memberikan gambaran visual dari proses kegiatan baik dalam suatu sistem bisnis, alur kerja, atau proses lainnya. Diagram ini fokus terhadap kegiatan yang dilakukan dan siapa yang bertanggung jawab dengan kinerja kegiatan tersebut. Terdapat beberapa proses dan simbol-simbol yang umum di gunakan di dalam *Activity diagram*, yaitu :

1. *Action State*

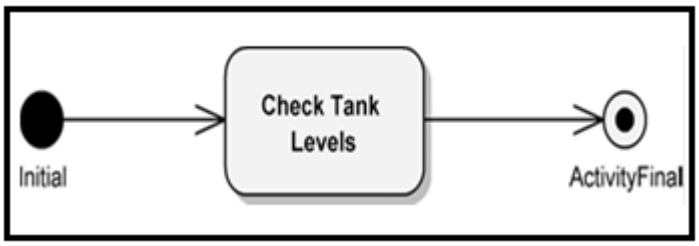


Gambar 2.18 Contoh *Action State*

Sumber Booch (2007:186)

Merupakan unsur pelaku dalam *Activity diagram* dimana kegiatan dapat memiliki banyak tindakan yang menunjukan jalannya suatu sistem.

2. *Initial* dan Final *Nodes*

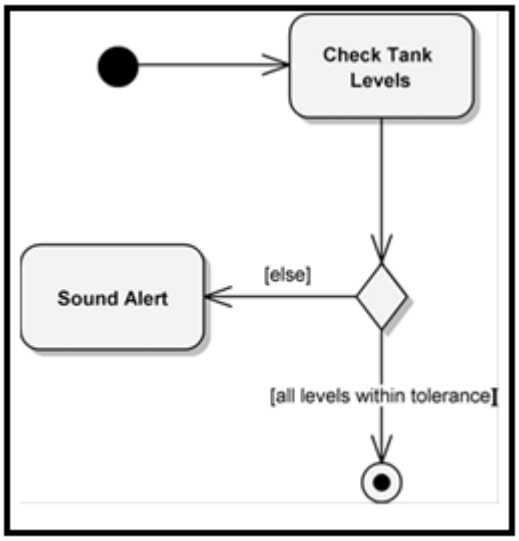


Gambar 2.19 Contoh *Initial* dan *Final Nodes*

Sumber Booch (2007:187)

Initial nodes merupakan aktifitas awal dari suatu sistem sedangkan *Final nodes* merupakan hasil akhir dari suatu sistem.

3. *Decision* dan *Merge Node*



Gambar 2.20 Contoh *Decision* dan *Merge Node*
Sumber Booch (2007:187)

Decision dan *Merge node* mengendalikan aliran pada *Activity diagram* dimana setiap node di gambarkan berbentuk berlian dengan panah masuk dan keluar. *Decision node* memiliki satu arah panah masuk dan beberapa panah keluar.

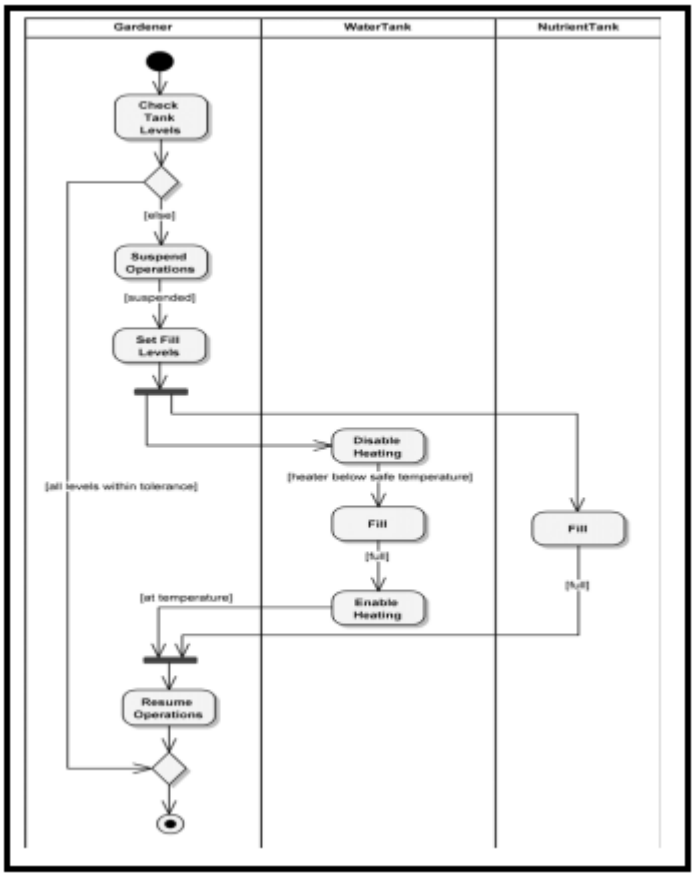
4. Partisi

   Dalam suatu *Activity diagram* dapat di kelompokkan dengan menggunakan partisi. Tujuannya ialah untuk menunjukan di mana tanggung jawab yang harus dilakukan dalam suatu kegiatan.

5. *Forks, Joins,* dan *Concurrency*

   Fork dan *Join nodes* di analogikan dengan *Decision* dan *Merge node* dimana *fork* memiliki satu aliran masuk dan beberapa arus keluar sedangkan *Join* memiliki beberapa aliran masuk dan memiliki satu aliran keluar, tetapi dalam *Join* seluruh aliran yang

masuk harus diselesaikan terlebih dahulu sebelum aliran keluar dimulai.



Gambar 2.21 Contoh Activity Diagram

Sumber Booch (2007:189)