BAB II

LANDASAN TEORI

2.1 Pengertian Sistem

2.1.1 Bentuk Dasar Sistem

Suatu sistem sangatlah dibutuhkan dalam suatu perusahaan atau instansi pemerintahan, karena sistem sangatlah menunjang terhadap kinerja perusahaan atau instansi pemerintah, baik yang berskala kecil maupun besar. Supaya dapat berjalan dengan baik diperlukan kerjasama diantara unsur-unsur yang terkait dalam sistem tersebut.

"*Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran yang tertentu"*. **Jogiyanto** (2005:1)

Masih dalam buku 'Analisis dan Desain sistem informasi' karangan jogiyanto menerangkan:

"Sistem adalah kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu". **Jogiyanto** (2005:2)

2.1.2 Karakteristik Sistem

Suatu sistem mempunyai karakteristik atau sifat–sifat yang tertentu, yaitu mempunyai komponen–komponen (*components*), batas sistem (*boundary*), lingkungan luar sistem (*environments*), penghubung (*interface*), masukan (*input*), keluaran (*output*), pengolah (*process*), dan sasaran (*objectives*), atau tujuan (*goal*).

a. Komponen Sistem

Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang artinya saling bekerjasama membentuk satu kesatuan. Komponen–komponen sistem atau elemen–elemen sistem dapat berupa suatu subsistem atau bagian–bagian dari sistem. Setiap sistem tidak peduli betapapun kecilnya, selalu mengandung komponen–komponen atau subsistem–subsistem.

b. Batas Sistem

Batas sistem (*boundary*) merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batas sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan. Batas suatu sistem menunjukkan ruang lingkup (*scope*) dari sistem tersebut.

c. Lingkungan Luar Sistem

Lingkungan luar (*environment*) dari suatu sistem adalah apapun diluar batas dari sistem yang mempengaruhi operasi sistem. Lingkungan luar sistem dapat bersifat menguntungkan dan dapat juga bersifat merugikan sistem tersebut. Lingkungan luar yang menguntungkan merupakan energi dari sistem dan dengan demikian harus tetap dijaga dan dipelihara. Sedangkan lingkungan luar yang merugikan harus ditahan dan dikendalikan, kalau tidak maka akan mengganggu kelangsungan hidup dari sistem.

d. Penghubung Sistem

Penghubung (*interface*) merupakan media penghubung antara satu sistem dengan subsistem lainnya. Melalui penghubung ini memungkinkan sumber – sumber daya mengalir dari satu subsistem ke subsistem yang lainnya. Dengan

penghubung satu subsistem dapat berintegrasi dengan subsistem yang lainnya membentuk satu kesatuan.

e. Masukan Sistem

Masukan (*input*) adalah energi yang dimasukkan kedalam sistem. Masukan dapat berupa masukan perawatan (*maintenance input*) dan masukan sinyal (*signal input*). Maintenance input adalah energi yang dimasukkan supaya sistem tersebut dapat beroperasi. Signal input adalah energi yang di proses untuk didapatkan keluaran.

f. Keluaran Sistem

Keluaran (*output*) adalah hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembangunan. Keluaran merupakan masukan untuk subsistem yang lain atau kepada supra sistem.

g. Pengolah Sistem

Suatu sistem dapat mempunyai suatu bagian pengolah yang akan merubah masukan menjadi keluaran. Suatu sistem produksi akan mengolah masukan berupa bahan baku dan bahan – bahan yang lain menjadi keluaran berupa barang jadi.

h. Sasaran Sistem

Suatu sistem pasti mempunyai tujuan (*goal*) atau sasaran (*objective*). Kalau suatu sistem tidak mempunyai sasaran, maka operasi sistem tidak akan ada gunanya. Sasaran dari sistem sangat menentukan sekali masukan yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem.

2.1.3 Klasifikasi Sistem

Sistem dapat diklasifikasikan dari beberapa sudut pandang, diantaranya adalah sebagai berikut :

a. Sistem terbuka dan tertutup

Sebuah sistem dikatakan terbuka bila aktivitas didalam sistem tersebut dipengaruhi oleh lingkungannya, sedangkan suatu sistem dikatakan tertutup bila aktivitas–aktivitas didalam sistem tersebut tidak terpengaruhi oleh perubahan yang terjadi di lingkungannya.

b. Sistem buatan manusia dan Sistem Alamiah

Suatu sistem diklasifikasikan berdasarkan asalnya, sistem tersebut bisa diklasifikasikan sebagai sistem yang ada secara alamiah (buatan Tuhan) atau buatan manusia.

c. Sistem Tertentu dan Tak Tentu

Suatu sistem dapat pula diklasifikasikan berdasarkan tertentu dan tak tentu. Sebuah sistem yang tertentu, misalkan sistem listrik ditempat dimana kita tinggal yang dipenuhi oleh arus listrik yang tetap dan dapat diukur. Sstem tak tentu, misalnya organisasi perusahaan merupakan sistem yang tidak dapat dipastikan kinerjanya.

d. Sistem fisik dan abstrak atau non fisik

Sistem dapat dilihat dari wujudnya. Kendaraan bermotor bukan hanya merupakan sistem yang ada secara fisik. Organisasi perusahaan dan perguruan tinggi bukanlah merupakan organisasi yang dapat disentuh secara fisik. 12

2.2 Pengertian Informasi

2.2.1 Konsep Dasar Informasi

Dalam manajemen, informasi merupakan data yang telah diproses sehingga mempunyai arti tertentu bagi penerimanya. Sumber dari informasi adalah Data, sedangkan data itu sendiri adalah kenyataan yang menggambarkan suatu kejadian, sedangkan kejadian itu merupakan suatu peristiwa yang terjadi pada waktu tertentu .dalam hal ini informasi dan data saling berkaitan.

"Informasi adalah data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya" . **Jogiyanto** (2005:8).

2.3 Sistem Informasi

2.3.1 Pengertian Sistem Informasi

"*Sistem informasi adalah suatu sistem didalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi ,bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan*". **Jogiyanto** (2005:11)

*"Menurut **Susanto** (2004 : 56) Sistem informasi adalah kumpulan dari sub-sub sistem baik fisik maupun non fisik yang saling berhubungan satu sama lain dan bekerja sama secara harmonis untuk mencapai satu tujuan yaitu mengolah data menjadi informasi yang berguna."*

2.3.2 Komponen Sistem Informasi

Komponen–komponen yang ada dalam sistem informasi meliputi beberapa blok, yaitu :

1. Blok masukan (input)

Blok masukan ini mewakili data yang masuk kedalam sistem informasi. Input disini termasuk metode-metode dan media untuk menangkap data yang akan dimasukan, yang dapat berupa dokumen - dokumen dasar.

2. Blok Model

Blok ini terdiri dari kombinasi prosedur, logika dan model matematika yang akan memanipulasi data input dan data yang tersimpan di basis data dengan cara yang sudah tertentu untuk menghasilkan keluaran yang diinginkan.

3. Blok keluaran (output)

Produk dari sistem informasi adalah keluaran yang merupakan informasi yang berkualitas dan dokumentasi yang berguna untuk semua tingkat manajemen serta semua pemakai sistem.

4. Blok Teknologi

Teknologi merupakan alat yang digunakan untuk menerima masukan, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan keluaran dan membantu pengendalian dari sistem secara keseluruhan. Teknologi terdiri dari 3 bagian utama, yaitu teknisi, perangkat lunak (*software*) dan perangkat keras (*hardware*).

5. Blok Basis Data.

Basis data merupakan kumpulan data yang saling berhubungan satu dengan yang lainnya, tersimpan diperangkat keras komputer, basis data diakses atau dimanipulasi dengan menggunakan paket perangkat lunak yang disebut data base manajemen sistem (DBMS).

6. Blok kendali

Beberapa pengendalian perlu dirancang dan diterapkan untuk meyakinkan bahwa hal-hal yang dapat merusak sistem bisa dicegah ataupun bila terlanjur terjadi kesalahan-kesalahan dapat langsung cepat diatasi.

2.4 *Message Short Service (SMS)*

2.4.1 Pengertian *Short Message Service (SMS)*

Short Message Service (SMS) adalah kemampuan untuk mengirim dan menerima pesan dalam bentuk teks dari dan kepada ponsel. Teks tersebut bisa terdiri dari kata-kata atau nomor atau kombinasi alphanumeric. *SMS* diciptakan sebagai standar pesan (*message*) oleh *ETSI* (*Europesan Telecommunication Standards Institute*), yang juga membuat standar GSM (*Global System for Mobile)* yang diimplementasikan oleh semua operator *Global System for Mobile* (*GSM*). *SMS* yang pertama dikirimkan pada Desember 1992 dari PC ke sebuah ponsel melalui jaringan *Global System for Mobile* (*GSM*) Vodafone di UK. Setiap Pesan maksimal terdiri dari 160 karakter jika menggunakan alphabet Latin, dan 70 karakter jika menggunakan alphabet non-Latin seperti huruf Arab atau China.

Pemanfaatan *SMS* saat ini sangat popular dibandingkan yang lainnya, disamping tarifnya murah tetapi juga mudah dan cepat dalam pemakaiannya sehingga sangat digemari oleh siapapun. Saat ini *SMS* juga mulai digunakan sebagai media penyebaran pesan ke banyak pengguna (broadcast) yang idenya diadopsi sebagai fasilitas peringatan dini (early warning system) yang digunakan sebagai peringatan akan kemungkinan terjadinya suattu bencana. *SMS* juga mulai dikembangkan sebagai perangkat pendaftaran elektronik (digital sign), perangkat penjawab umpan balik otomatis (auto responder), dan *SMS* dapat difungsikan sebagai alat control aktivasi (*remote controller activator*) ke dalam perangkat computer seperti untuk mematikan computer, menyalakan lampu, menyalakan sistem keamanan, mengetahui posisi dan lain-lain. Secara umum *SMS* telah dikembangkan untuk (Zakaria & Josef, 9) :

1. Sistem pemilihan (*polling*) dan jajak pendapat (*voting*).
2. Pengiriman banyak serentak (*broadcasting*)
    1. Pengumuman (*announcer*)
    2. Peringatan (*warning*)
    3. Pengingat (*reminder*)
3. Alat control aktivasi (*remote controller activator*)
4. Penjawab umpan balik otomatis (*auto responder*)
    1. informasi (*Information*)
    2. Pendaftaram bertingkat (*cascading digital sign*)
    3. Pendaftaran tersusun (*smart tag digital sign*)

5. Perangkat penguji jaringan (tester)

1. *Flood SMS/SMS Bomb*
2. *Floating SMS,* dan lain-lain.

*Menurut **riswan** (2006), Teknologi SMS memiliki banyak keunggulan-keunggulan tersebut antara lain :*

1. *Harga untuk mengirim sebuah SMS murah*
2. *Merupakan "deliver oriented service", artinya pesan akan selalu diusahakan untuk dikirimkan ke nomor tujuan. Jika suatu saat nomor tujuan sedang aktif atau diluar cakupan jaringan, maka pesan akan disimpan di SMSC server dan akan dikirimkan sesegera setelah nomor tujuan aktif kembali. Pesan juga akan tetap terkirim ke tujuan walaupun nomor tujuan sedang melakukan pembicaraan (sibuk).*
3. *Dapat dikirim ke banyak penerima sekaligus pada saat yang bersamaan.*
4. *Pesan dapat dikirimkan ke berbagai jenis tujuan, seperti e-mail, Ip ataupun aplikasi lain.*

Dari beberapa pernyataan diatas dapat diketahui bahwa *SMS* tidak hanya digunakan untuk keperluan bertukar informasi dengan teks saja, tetapi saat ini *SMS* sudah dikembangkan untuk berbagai keperluan yang dapat mempermudah manusia dalam berbagai keperluan seperti untuk mengirimkan gambar dan suara secara bersamaan dengan *SMS* yang dikenal dengan nama *Enhanced Message Service (EMS), Multimedia Message Service (MMS), Smart Messaging* bahkan telah mendukung pertukaran data tedefinisi melalui konsep OTA (*Over-The-Air*)

seperti bertukar buku telepon (*Phonebook*), bertukar jadwal (*note, schedule, reminder*), dan secara terbatas bertukar nada dering (*ring tones*).

2.4.2 Layanan Aplikasi SMS

*"layanan aplikasi pada SMS pada dasarnya memiliki karakteristik yang berbeda dengan aplikasi internet yaitu ukuran layar monitor yang berukuran kecil, serta keterbatasan jumlah karakter yang bisa dikirimkan". **Zakaria & Josef** (2006 : 34)*

Dari karakteristik tersebut dapat mendasari pada pengembangan aplikasi yang dapat menyediakan informasi secara singkat dan jelas dengan pengoperasian aplikasi yang mudah dan sederhana. Berdasarkan mekanisme distribusi pesan *SMS* oleh aplikasi *SMS,* terdapat empat macam mekanisme penghantaran pesan, yaitu :

1. *Pull,* yaitu pesan yang dikirimkan ke pengguna berdasarkan permintaan pengguna.
2. *Push - Event based,* yaitu pesan yang diaktivasi oleh aplikasi berdasarkan kejadian yang berlangsung.
3. *Push – Scheduled,* yaitu pesan yang diaktivasi oleh aplikasi berdasarkan waktu yang telah terjadwal.
4. *Push – Personal profile,* yaitu pesan yang diaktivasi oleh aplikasi berdasarkan profil dan preference dari pengguna.

2.4.3 Teknologi SMS

"Short Message Service (SMS) merupakan sebuah layanan yang banyak diaplikasikan pada sistem komunikasi tanpa kabel, memungkinkan dilakukannya

pengiriman pesan dalam bentuk alphanumeric antara terminal pelanggan atau antara terminal pelanggan dengan sistem eksternal seperti email, paging, voice mail, dan lain-lain." **Rosidi** (2004: 1)

Isu SMS pertama kali muncul di belahan Eropa pada sekitar tahun 1992 bersama sebuah teknologi komunikasi wireless yang saat ini cukup banyak penggunanya, yaitu GSM (Global Sistem For Mobile Communication).

Dipercaya bahwa message pertama yang dikirimkan menggunakan SMS dilakukan pada bulan Desember 1992, dikirimkan dari sebuah PC (Personal Computer) ke telephone mobile dalam jaringan GSM milik Vodafone Inggris. Perkembangannya kemudian merambah ke benua Amerika, dipelopori oleh beberapa operator komunikasi bergerak berbasis digital seperti BellSouth Mobility, PrimeCo, Nextel dan beberapa operator lain teknologi digital yang digunakan bervariasi dari yang berbasis GSM, TDMA (Time Division Multiple Access), hingga CDMA (Code Division Multiple Access).

Dalam sistem SMS, mekanisme utama yang dilakukan dalam sistem adalah melakukan pengiriman short message dari satu terminal pelanggan ke terminal yang lain. Hal ini dapat dilakukan berkat adanya sebuah entitas dalam sistem SMS yang bernama SMSC (Short Message ServiceCenter) atau disebut juga MC (Message Center). Layanan SMS merupakan sebuah layanan yang bersifat nonreal time di mana sebuah short message dapat di kirim ke suatu tujuan, tidak peduli apakah tujuan tersebut aktif atau tidak. Bila dideteksi bahwa tujuan tidak aktif, maka sistem akan menunda pengiriman ke tujuan hingga tujuan aktif kembali.

2.4.4 *SMS Gateway*

"Istilah gateway, bila dilihat pada kamus Inggris-Indonesia diartikan sebagai pintu gerbang. Namun pada dunia komputer, gateway dapat berarti juga sebagai jembatan penghubung antara satu sistem dengan sistem lain yang berbeda, sehingga dapat terjadi suatu pertukaran data antara sistem tersebut. Dengan demikian, SMS gateway dapat diartikan sebagai suatu penghubung untuk lalulintas data-data SMS, baik yang dikirimkan maupun yang diterima." **Yunianto** (2006:25)

Ada dua cara mengirimkan pesan dari kompter / mobile phone:

1. Sambungkan mobile phone atau GSM/GPRS modem ke kompter/PC. Kemudian gunakan komputer/PC dan AT *Command*s untuk menginstruksikan mobile phone atau GSM/GPRS modem untuk mengirimkan pesan sms.
2. Sambungkan komputer/PC ke sms center (SMSC) atau sms *gateway* dari perusahaan wireless atau sms service provider. Lalu kirimkan pesan sms dengan menggunakan protokol yang disupport oleh SMSC atau sms *gateway*.

Ada tiga cara untuk menerima pesan sms dengan menggunakan komputer / pc:

1. Sambungkan mobile phone atau *GSM/GPRS* modem ke komputer/pc. Kemudian gunakan komputer / pc dan AT Commands untuk menerima sms dari mobile phone atau *GSM/GPRS* modem.
2. Dapatkan akses ke sms center (SMSC) atau sms gateway dari perusahaan wireless. Pesan sms yang diterima akan dikirimkan ke komputer / pc dengan menggunakan protokol yang disupport oleh *SMSC* atau *sms gateway*.

3. Dapatkan akses ke sms gateway dari sms service provider. Pesan sms yang diterima akan dikirimkan ke komputer / pc dengan menggunakan protokol yang disupport oleh *sms gateway.*

"Sistem minimal dari SMS Gateway adalah sebuah Personal Computer yang dilengkapi dengan aplikasi SMS Gateway, perangkat komunikasi berbasis selluler (Telpon Selluler atau Modem GSM/CDMA) yang terhubung dengan PC. Aplikasi SMS Gateway adalah sebuah perangkat lunak yang mampu membaca informasi SMS yang diterima dan melakukan pengelolaan sesuai kebutuhan." **dewanto** (2007:17)

Untuk koneksi dengan telpon selluler menggunakan perintah AT Command melalui port I/O (serial/USB) yang mendukung GSM Modem ataupun berbagai merk telpon selluler. Untuk aplikasi pembacaan data SMS menggunakan konversi PDU Format dan metoda parsing untuk memisahkan kode-kode transaksi. Pembacaan dan pengelolaan pesan berikutnya menggunkan sebuah aplikasi berbasis Sistem Informasi Manajemen.

Istilah *gateway* dapat diartikan sebagai pintu gerbang. Namun pada dunia komputer, *gateway* dapat diartikan sebagai jembatan penghubung antara satu sistem dengan sistem yang lain, sehingga dapat terjadi pertukaran data antar sistem tersebut. Dengan demikian, *SMS Gateway* dapat diartikan sebagai pintu gerbang atau jalur bagi penyebaran informasi dengan menggunakan *Short Message Service (SMS)* Dengannya, kita dapat menyebarkan pesan ke banyak nomor sekaligus secara otomatis dan cepat. Dengan menggunakan program tambahan yang dapat dibuat sendiri, si pengirim pesan dapat lebih fleksibel dalam

mengirim/menerima *Short Message Service (SMS)* sesuai dengan aturan yang ditetapkan, seperti sistem untuk *polling* misalnya. Cara kerja *SMS Gateway* : *SMS* dikirim pemilih ke nomor yang telah ditentukan, isi *SMS* merupakan Format yang telah ditentukan sebelumnya. *SMS* diterima oleh *Handphone* yang bekerja sebagai *SMS Gateway* kemudian akan dikirimkan ke sistem, sistem akan merespon *SMS* tersebut. Jika Format *SMS* benar maka *Handphone* yang bertindak sebagai *SMS Gateway Server* akan membalas *SMS* tersebut bahwa *SMS* telah ditampung begitu juga sebaliknya. Kemudian *SMS* tersebut ditampung dalam *database* dan akan ditampilkan kedalam *Website*.



Gambar 2.1 Cara Kerja *SMS Gateway*

Sumber : http://master-digital.blogspot.com/2009/09/membuat-sendiri-sms-gateway.html/18 April 2012

2.5 Perangkat Keras Yang Digunakan

Menurut **Sutono** (2010:1) ada beberapa komponen perangkat keras yang mempengaruhi sistem informasi adalah :

1. Motherboard adalah sirkuit utama dimana komponen-komponen lainnya seperti *CPU, RAM,* kartu grafis, kartu suara, kartu jaringan, dan sebagainya ditancapkan.
2. *CPU (Central Processing Input)* adalah otak komputer, naik jika komputer tersebut berfungsi sebagai work station maupun *server*. Tanpa *CPU* tidak mungkin kita mengharapkan suatu pekerjaan akan dapat dikerjakan oleh komputer.
3. Memory (*RAM / Random Access Memory*) setiap aplikasi pada saat pemrosesan oleh *CPU* dilakukan, akan diletakkan dalam memori internal (*RAM*) komputer yang bersifat sementara (*volatile*) agar pemrosesan berlangsung dengan lebih cepat.
4. Layar Monitor bagian untuk menampilkan program.
5. *Hardisk* Media penyimpanan data.
6. Peripheral Tambahan *(Mouse dan disk drive),* mouse adalah alat bantu petunjuk, dan disk drive adalah media penyimpanan lain selain hardisk.
7. Modem alat penghubung komputer dengan jalur telepon.
8. *Printer* alat cetak dokumen.

2.6 Perangakat Lunak Yang Digunakan

Komputer tidak berguna tanpa keberadaan perangkat lunak (*software*). Komputer bekerja atas dasar instruksi. Sekumpulan instruksi diberikan untuk mengendalikan perangkat keras *komputer*. Sekumpulan instruksi inilah yang

dikenal dengan sebutan program atau *program komputer*. Secara lebih umum, *program komputer* inilah yang disebut perangkat lunak.

Perangkat lunak dikelompokkan menjadi dua, yaitu program aplikasi dan program sistem. Program aplikasi adalah program yang dibuat oleh pemakai yang ditujukan untuk melakukan suatu tugas khusus. Program seperti ini biasa dikelompokan menjadi dua, yaitu program aplikasi serbaguna dan program aplikasi spesifik.

Sedangkan, program sistem adalah program yang digunakan untuk mengontrol sumber daya komputer, seperti *CPU* dan piranti masukan/keluaran. Kedudukan program ini adalah sebagai perantara antara program aplikasi dan perangkat keras komputer. Itulah sebabnya, peran program sistem sering kali tidak terlihat secara langsung. Program sistem dapat dikelompokkan menjadi tiga golongan, yaitu program pengendalian sistem, program pendukung sistem dan program pengembangan sistem.

2.6.1 *CodeIgniter*

Saat ini *Framework PHP* yang paling populer digunakan adalah *CodeIgniter*. Tidak hanya di Indonesia saja, tetapi di luar negeri sana juga sangat pupuler. Makanya kita tidak akan heran jika akhir-akhir ini, semakin banyak perusahaan yang mencari calon *programmer* yang mahir menggunakan *CodeIgniter* dalam membangun aplikasi berbasis *website*.

CodeIgniter adalah aplikasi open source yang berupa *framework* dengan model *MVC* (Model, View, Controller) untuk membangun websitedinamis dengan menggunakan *PHP*. *CodeIgniter* memudahkan *developer* untuk membuat aplikasi

webiste dengan cepat dan mudah dibandingkan dengan membuatnya dari awal. *CodeIgniter* dirilis pertama kali pada 28 Februari 2006. Versi stabil terakhir 2.1.0 yang dirilis pada 28 November 2011.

Model View Controller merupakan suatu konsep yang cukup populer dalam pembangunan aplikasi web, berawal pada bahasa pemrograman Small Talk, *MVC* memisahkan pengembangan aplikasi berdasarkan komponen utama yang membangun sebuah aplikasi seperti manipulasi data, user interface, dan bagian yang menjadi kontrol aplikasi. Terdapat 3 jenis komponen yang membangun suatu *MVC* pattern dalam suatu aplikasi yaitu :

1. View, merupakan bagian yang menangani presentation logic. Pada suatu aplikasi web bagian ini biasanya berupa file template *HTML*, yang diatur oleh controller. View berfungsi untuk menerima dan merepresentasikan data kepada user. Bagian ini tidak memiliki akses langsung terhadap bagian model.
2. Model, biasanya berhubungan langsung dengan database untuk memanipulasi data (insert, update, delete, search), menangani validasi dari bagian controller, namun tidak dapat berhubungan langsung dengan bagian view.
3. Controller, merupakan bagian yang mengatur hubungan antara bagian model dan bagian view, controller berfungsi untuk menerima request dan data dari user kemudian menentukan apa yang akan diproses oleh aplikasi.

Ada beberapa kelebihan *CodeIgniter* (*CI*) dibandingkan dengan *Framework PHP* lain :

1. Performa sangat cepat. Salah satu alasan tidak menggunakan *framework* adalah karena eksekusinya yang lebih lambat daripada *PHP* from the scracth, tapi *CodeIgniter* sangat cepat bahkan mungkin bisa dibilang *CodeIgniter* merupakan *framework* yang paling cepat dibanding *framework* yang lain.
2. Konfigurasi yang sangat minim (nearly zero configuration). Tentu saja untuk menyesuaikan dengan database dan keleluasaan routing tetap diizinkan melakukan konfigurasi dengan mengubah beberapa file konfigurasi seperti database.*PHP* atau autoload.*PHP*, namun untuk menggunakan *CodeIgniter* dengan setting standard, anda hanya perlu merubah sedikit saja file pada folder config.
3. *CodeIgniter* memiliki banyak komunitas. Dengan banyaknya komunitas *CI* ini, memudahkan kita untuk berinteraksi dengan yang lain, baik itu bertanya atau teknologi terbaru.
4. Dokumentasi yang sangat lengkap. Setiap paket instalasi *CodeIgniter* sudah disertai user guide yang sangat bagus dan lengkap untuk dijadikan permulaan, bahasanya pun mudah dipahami.
5. *Programmer CodeIgniter* paling banyak dicari perusahaan. Saat ini, *Programmer* yang menguasai *CodeIgniter* dalam mebangun aplikasi website sangat di cari-cari oleh banyak perusahaan. Hal ini penulis bisa lihat dari lowongan kerja untuk *Programmer* di JobsDB.com dan juga karir.com,

rata-rata mencari *Programmer* yang memiliki keahlian dalam *CodeIgniter*. Selain itu dari pengalaman saya sebagai *Programmer* dalam 4 Tahun terakhir ini, saya bisa merasakan dengan menggunakan *CodeIgniter* sebagai tools saya dalam mengembangkan aplikasi web, kemampuan saya menjadi meningkat drastis. Baik dari segi sekuriti, flow data, bahkan struktur programnya menjadi sangat tertata rapi. Sehingga memudahkan dalam maintenace dan pengembangan aplikasi.

2.6.2 *Gammu*

Gammu merupakan salah satu pustaka atau *library open-source* yang dibuat sebagai *gateway* antara *hand-phone* dengan perangakat komputer. Pengembangan *Gammu* awalnya dari pendahuluannya yaitu Gnooki yang dari segi konsep masih sangat sederhana dan rumit. Namun pada *Gammu* proses instalasi dan penerapan pembangunan aplikasi *SMS* semakin mudah. Ini merupakan jasa dari beberapa *developer* yang telah membangun *Gammu*.

Gammu adalah semacam service yang disediakan untuk membangun aplikasi yang berbasis *SMS Gateway*. Setelah kita sukses membangun *Gammu*, maka kita bisa membangun aplikasi *SMS Gateway* dengan bahasa pemrograman atau platform apapun, baik itu web based dengan *PHP* maupun ASP atau apapun, dan juga desktop dengan menggunakan *Delphi*, *VB* atau lainnya.

Kelebihan dari *Gammu* ini adalah merupakan software opensource yang tersedia dalam bentuk source code. Selain itu *Gammu* juga tersedia untuk *Operating Sistem Windows* maupun *Linux*.

Terkait dengan masalah kemampuan, *Gammu* tidak perlu diragukan lagi karena selain bisa untuk mengirim/menerima *SMS*, *Gammu* juga dapat mengirim/menerima *MMS*, *backup atau restore phonebook*, serta upload/download file ke *handphone*. Selain itu *Gammu* juga support untuk lebih dari 400 jenis *handphone*.

Contoh script *PHP* sederhana untuk mengirim pesan *SMS* yang terintegrasi dengan *Gammu* :

send.*PHP*

```
<?PHP
$noTujuan = $_POST['noHP'];
$message = $_POST['msg'];
exec('c:\Gammu\Gammu-SMSd-inject.exe -c c:\Gammu\SMSdrc EMS '.$noTujuan.' -text "'.$message.'"');
?>
```

Perintah exec() dalam *PHP* digunakan untuk menjalankan perintah command *DOS* dalam hal ini adalah perintah command inject nya *Gammu*. Karena command inject *Gammu* pada script di atas menggunakan format EMS maka memungkinkan kita mengirim *SMS* dengan jumlah karakter lebih dari 160 buah.

2.6.3 Pengenalan *Dreamweaver 8*

Dreamweaver merupakan salah satu *software* dari kelompok Macromedia yang banyak digunakan untuk mendesain situs *web*. Adapun macromedia dreamweaver itu sendiri adalah sebuah *HTML* editor profesional yang berfungsi

untuk mendesain secara visual dan mengelola situs *website* maupun halaman *website.*

Dreamweaver 8 merupakan versi terbaru yang memiliki performa yang lebih baik dan memiliki tampilan yang memudahkan siapapun untuk membuat halaman *web*, baik ketika sedang bekerja dalam jendela desain maupun dalam jendela kode rumus. *Dreamweaver 8* didukung untuk pengembangan menggunakan *CSS, XML* dan *RSS* serta kemudahan lain yang diperlukan. Bisa mendapatkan kecepatan dengan memaksimalkan penggunaan *XML*, tombol-tombol tampilan untuk memadukan bagian kedalam pekerjaan dan mengganti sensasi dari *XML* ke bahasa *HTML*. Siapapun juga dapat memadukan sumber data *XML*, seperti bagian *RSS* ke halaman *web* hanya menggunakan cara klik tahan geser mouse secara manual dan dapat bekerja pada tampilan kode untuk mengembangkan perubahan dengan membaca kode pertunjuk dari *XML* dan *XSLT*.

Adapun fasilitas yang ada dalam *Dreamweaver 8,* sebagai berikut :

1. *Dreamweaver 8* masih dapat bekerja meskipun file sedang di *upload* ke jaringan.
2. Desain dapat dikontrol dengan fasilitas baru *Zoom*. *Zoom In* memungkinkan untuk meneliti gambar atau pekerjaan dengan *layout table* yang lebih komplek atau luas. Adapun *Zoom Out* untuk menampilkan contoh hasil tampilan halaman.
3. Tampilan *Guides* dengan satuan pixel yang sempurna sangat membantu mendapatkan ukuran yang lebih akurat dalam mengukur layout tampilan.

4. *Coding Toolbar*, toolbar baru ini ada pada jendela kode yang digunakan untuk mempermudah pekerjaan pada jendela tersebut.
5. Tampilan baru dalam *Dreamweaver 8* membuat pekerjaan lebih cepat selesai.
6. *Paste* special, dapat mengambil teks dari *Microsoft Word* beserta formatnya.

2.6.4 Apache

Apache adalah sebuah HTTP *server*. Apache dibangun pertama kalinya berdasarkan pada kode-kode dan ide-ide yang terdapat pada *HTTP server* yang terkenal pada saat ini, yaitu *NCSA* httpd 1.3, pada awal 1995.

Pada saat ini Apache termasuk *web server* yang paling banyak digunakan, hal ini terjadi bukan semata-mata karena Apache diedarkan secara gratis. Namun juga karena kemampuannya, secara fungsional, tingkat efisien dan kecepatannya yang dapat dibandingkan dengan *server HTTP* UNIX lainnya. Selain diedarkan secara gratis dan *open source*, pengembangan Apache juga dilakukan oleh banyak *programmer* secara sederhana.

2.6.5 Sistem Database *MySQL*

Menurut **Madcom** (2004 : 177) MySQL merupakan software database yang ternasuk paling populer di lingkungan Linux, kepopuleran ini karena ditunjang performansi query dari databasenya yang saat itu bisan dikatakan paling cepat dan jarang bermasalah.Berangkat darisoftware yang shareware MySQL populer, kini mulai versi 3.23 MySQL menjadi software open source yang berarti free. MySQL dapat digunakan untuk kepentingan komersial atau pun personal (non profit).bMySQL telah tersedia juga di lingkungan windows, software

MySQL di lingkungan Windows dipasang pada direktori c:\mysql. c:\mysql\bin adalah direktori yang berisi daftar modul executable dari software MySql. PHP untuk Windows secara default telah mendukung MySQL.

2.6.6 *PHP* (*PHP Hypertext Preprocessor*)

PHP adalah *script* untuk pemrograman *web*, *PHP* dapat digunakan bersama – sama dengan kode *HTML (embedded).* Sehingga hampir mirip dengan *javascript*, bedanya dengan *javascript* adalah *javascript* akan diterjemahkan oleh internet *client (browser),* sedangkan *script PHP* diterjemahkan di pihak *server*. Karenanya *PHP* juga dikenal sebagai *server side script*. Halaman hasil proses di *server,* diterima oleh *client* internet sebagai *HTML* biasa, dan kode – kode *PHP* tidak akan bisa terbaca oleh *client*. Hal ini juga yang membedakan *PHP* dengan *javascript*.

Script PHP ini mulai bekerja saat adanya *request PHP file* tersebut dari *Web browser*. *Web server* kemudian akan memanggil *file* tersebut, dan akan membaca semua *request* dari atas sampai bawah. Mengeksekusi beberapa *script command,* dan mengirimkan hasilnya dalam sebuah halaman *HTML* melalui *browser*.

Sebuah *file PHP* adalah *text file* dengan *extension* ***.PHP*** yang berisi kombinasi dari :

- ✓ *Text*
- ✓ *HTML Tags*
- ✓ *PHP Script commands*
- ✓ Memungkinkan untuk bahasa script yang lain.’

Penulisan *script PHP* ini diawali dengan **"<?"** dan ditutup dengan **"?>"**. Dalam pembuatan halaman *Web* untuk hasil yang lebih baik bisa digunakan lebih dari satu bahasa penulisan *script*. Dan untuk menandainya digunakan sintak penulisan sebagai berikut :

<? *HTML* dan variable SQL ?> atau jika ada *script* yang lain dapat ditambahkan penulisannya, <script language=Scripting Language">

Sintak penulisan *script* dan logika berikutnya sesuai dengan bahasa pemrograman yang digunakan. Jadi dalam sebuah *script PHP* bisa terdapat lebih dari satu bahasa *script* yang digunakan.

2.6.7 *HTML* (HyperText Markup Language)

HTML adalah suatu bahasa pemrograman yang dipergunakan untuk menyusun dan membentuk suatu dokumen agar dapat ditampilkan pada program *browser WWW (world wide web)* dalam bentuk yang dikehendaki oleh pembuatnya.

HTML juga dipergunakan untuk menciptakan *hypertext link* atau hubungan antara teks dan dokumen lain atau juga bagian dari dokumen lain, *HTML* hanya berupa file teks yang bisa dibaca dengan suatu editor teks biasa, sesuai dengan namanya bahasa ini menggunakan tanda (*markup*) untuk menandai perintah-perintahnya.

2.6.8 Pengenalan *APPSERV*

Appserv merupakan sebuah *tool* yang menyediakan beberapa paket perangkat lunak ke dalam satu buah paket. Dengan menginstal *Appserv* tidak perlu melakukan instalasi dan konfigurasi *web server* Apache, *PHP* dan *MySQL*

secara manual. *Appserv* akan menginstalasi dan konfigurasikannya secara otomatis.

Appserv merupakan pengembangan dari *LAMP* (*Linux* Apache, *MySQL*, *PHP* and *PERL*), APPSERV ini merupakan *project* non-profit yang di kembangkan oleh Apache Friends yang didirikan Kai Oswalad‟ Seider dan Kay Vogelgesang pada tahun 2002, *project* mereka ini bertujuan untuk mempromosikan penggunaan Apache *web server*.

2.7 Analisis Sistem

Tahap analisis system merupakan tahap yang kritis dan sangat penting, karena kesalahan didalam tahap ini akan menyebabkan juga kesalahan ditahap selanjutnya. Istilah analisis system ini memang tepat, karena memang itulah yang akan dikerjakan oleh analisis system dalam tahap ini, yaitu menganalisis system untuk menemukan kelemahan-kelemahannya sehingga dapat diusulkan perbaikannya.

2.7.1 Pengertian Analisis Sistem

Menurut **Jogiyanto** (2005:1) *analisis sistem adalah penguraian dari suatu system informasi yang utuh kedalam bagian-bagian komponennya dengan maksud untuk menguidentifikasi dan mengevaluasi permasalahan-permasalahan, kesempatan-kesempatan, hambatan-hambatan yang terjadi dan kebutuhan-kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan*.

2.7.2 Langkah-langkah Analisis Sistem

Di dalam tahap analisis terdapat langkah-langkah dasar yang harus dilakukan oleh analisis sistem, lang-langkah tersebut adalah :

1. *Identify* (mengidentifikasi masalah)

Mengidentifikasi masalah merupakan langkah pertama yang dilakukan dalam tahap analisis sistem. Masalah dapat didefinisikan sebagai suatu pertanyaan yang diinginkan untuk dipecahkan. Masalah inilah yang menyebabkan sasaran dari sistem tidak dapat dicapai. Oleh karena itulah pada tahap analisis sistem, tahap pertama yang harus dilakukan adalah mengidentifikasi masalah.

2. *Understand* (memahami kerja dari sistem yang ada)

Langkah kedua dari tahap analisis sistem adalah memahami kerja dari sistem yang ada. Langkah ini dapat dilakukan dengan mempelajari secara terinci bagaimana sistem yang ada beroprasi. Untuk mempelajari operasi dari sistem ini diperlukan data yang dapat diperoleh dengan cara melakukan penelitian.

3. *Analyze*, yaitu menganalisis sistem

Langkah ini dilakukan berdasarkan data yang telah diperoleh dari hasil penelitian yang telah dilakukan.

4. *Report,* membuat laporan hasil analisis

Setelah analisis sistem selesai dilakukan, langkah terakhir yang harus dilakukan oleh analis sistem adalah membuat laporan hasil analisis.

2.7.3 Perancangan Sistem

Setelah tahap analisis sistem selesai dilakukan, maka analis sistem telah mendapatkan gambaran dengan jelas apa yang harus dikerjakan. Tahap selanjutnya yang perlu dilakukan adalah merancang sistem (*system design*).

2.7.4 Pengertian Perancangan Sistem

Menurut Jogiyanto (2005:3) perancangan sistem dapat diartikan sebagai berikut:

1. Tahap setelah analisis dari siklus pengembangan sistem
2. Pendefinisian dari kebutuhan-kebutuhan fungsional
3. Persiapan untuk rancangan bangun *implementasi*
4. Menggambarkan bagaimana suatu sistem dibentuk
5. Yang dapat berupa penggambaran, perencanaan dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah kedalam satu kesatuan yang utuh dan berfungsi
6. Termasuk menyangkut mengkonfigurasi dari komponen-komponen perangkat lunak dan perangkat keras dari suatu sistem.

2.7.5 Tujuan Perancangan Sistem

Menurut **Jogiyanto** (2005:6) tahap perancangan sistem mempunyai dua tujuan utama, yaitu sebagai berikut :

1. Untuk memenuhi kebutuhan kepada pemakai sistem.
2. Untuk memberikan gambaran yang jelas dan rancangan bangunan yang lengkap kepada pemrogram komputer dan ahli-ahli teknik lainnya yang terlibat.

Untuk mencapai tujuan ini, analisis sistem harus dapat mencapai sasaran-sasaran sebagai berikut:

1. Perancangan sistem harus berguna, mudah dipahami dan nantinya mudah dipergunakan. Ini berarti bahwa data harus mudah ditangkap, metode-metode harus mudah diterapkan dan informasi harus mudah dihasilkan serta mudah dipahami dan digunakan.
2. Perancangan sistem harus dapat mendukung tujuan utama perusahaan sesuai dengan yang telah didefinisikan pada tahap analisis sistem.
3. Perancangan sistem harus efisien dan efektif untuk dapat mendukung pengolahan transaksi, pelaporan manajemen dan mendukung keputusan yang akan dilakukan oleh manajemen, termasuk tugas-tugas yang lainnya yang tidak dilakukan oleh computer
4. Perancangan sistem harus dapat mempersiapkan rancangan bangun yang terinci untuk masing-masing komponen dari sistem informasi yang meliputi data dan informasi, simpanan data, metode-metode, prosedur-prosedur, orang-orang, perangkat keras, perangkat lunak dan pengendalian intern.

2.7.6 Pendekatan Berorientasi Objek

Menurut **Adi Nugroho** (2005 : 4) yang dimaksud dengan pendekatan berorientasi objek adalah.

Suatu cara baru dalam berpikir serta berlogika dalam menghadapi masalah-masalah yang akan dicoba-atasi dengan bantuan komputer, mencoba melihat permasalahan lewat pengamatan dunia nyata dimana setiap objek adalah entitas tunggal yang memiliki kombinasi struktur data dan fungsi tertentu.

Suatu cara pengembangan perangkat lunak dan sistem informasi berdasarkan objek-objek yang ada di dunia nyata.

Teknologi berorientasi objek menganalogikan sistem aplikasi seperti kehidupan nyata yang didominasi objek. Berikut prinsip dasar dari pemograman berorientasi objek yang terdiri dari :

1. Abstraksi : memfokuskan pada karakteristik objek
2. Kelas : bagian yang dapat menciptakan objek
3. Enkapsulasi (Pembungkusan) : Menyembunyikan banyak hal yang terdapat dalam objek yang tidak perlu diketahui objek lain
4. Generlaisasi (Spesialisasi) : pembagian kelas-kelas yang berperilaku sama.
5. Polimorfisme : fungsi yang sama dapat diterapkan dan dimiliki oleh kelas-kelas berlainan.

2.7.7 UML *(Unified Modeling Language)*

UML dikeluarkan oleh OMG (*Object Management Group, Inc)* yaitu organisasi internasional yang dibentuk pada 1989, terdiri dari perusahaan sistem informasi, *software developer* dan para *user* sistem komputer.

Beberapa konsep dasar dalam OOP (*Object Oriented Programming*) menurut Martin Flower :

1. *Object*

   Objek (*object*) adalah “benda”, secara fisik atau konseptual, yang dapat kita temui disekeliling kita. Hardware, software, dokumen, manusia, dan bahkan konsep semuanya adalah contoh objek. Untuk kepentingan

memodelkan perusahaannya, seorang kepala eksekutif akan melihat karyawan, gedung, divisi, dokumen, dan keuntungan sebagai objek. Seorang teknisi mesin akan melihat ban, pintu, mesin, laju tertinggi, dan banyaknya bahan bakar sebagai objek. dan, seorang *software engineer* akan memandang tumpukan, antrian, jendela (window), dan check box sebagai objek. Sebuah objek memiliki keadaan sesaat (state) dan perilaku (behavior). State dari sebuah objek adalah kondisi objek tersebut atau himpunan dari keadaan yang menggambarkan objek tersebut. Sebagai contoh, bola lampu adalah objek, dan salah satu keadaan nyala atau tidak adalah state dari objek bola lampu tersebut. Salah satu state dari objek jam adalah waktu saat ini. State dinyatakan dengan nilai dari atribut (attribute) objeknya. Atribut adalah nilai internal suatu objek yang mencerminkan antara lain karakteristik objek, kondisi sesaat, koneksi dengan objek lain, dan identitasnya. Perubahan state dicerminkan oleh perilaku (behavior) objek tersebut.

Behavior suatu objek mendefinisikan bagaimana sebuah objek bertindak (beraksi) dan memberi reaksi. Behavior ditentukan oleh himpunan semua atau beberapa operasi yang dapat dilakukan dalam objek itu sendiri. Behavior dari sebuah objek dicerminkan oleh interface, service, dan method dari objek tersebut. Interface adalah pintu untuk mengakses *service* objek. Service adalah fungsi yang bisa diemban objek. Method adalah mekanisme internal objek yang mencerminkan perilaku (behavior)

objek tersebut. Sebagai contoh, jika printer sebuah merupakan perilaku (behavior) atau service-nya mencetak apapun yang dia terima.

2. *Class*

   Kelas (class) adalah definisi umum (pola, template atau cetak biru) untuk himpunan objek sejenis. Kelas menetapkan spesifikasi perilaku (behaviors) dan atribut objek-objek tersebut. Class adalah keniskalan (abstraksi) dari entitas dalam dunia nyata. Objek adalah "contoh" (instance) dari sebuah kelas. Sebagai contoh, atribut untuk kelas binatang adalah berkaki empat dan memiliki ekor. Perilakunya adalah tidur dan makan. Contoh (instance) yang mungkin dari kelas binatang ini adalah kucing, gajah, dan kuda.

3. *Inheritance*

   Seperti yang sudah diuraikan di atas, objek adalah contoh/ instance dari sebuah class. Hal ini mempunyai konsekuensi yang penting yaitu sebagai instande sebuah class, sebuah objek mempunyai semua karakteristik dari classnya. Inilah yang disebut dengan inheritance (pewarisan sifat). Dengan demikian apaun attribute dan operation dari class akan dimiliki pula oleh semua objek yang diinherit/ diturunkan dari class tersebut. Sifat ini tidak hanya berlaku untuk objek terhadap class, akan tetapi juga berlaku untuk class terhadap class lainnya.

4. *Polimorphisme*

   Polimorphisme adalah konsep yang handal bagi pengembangan perangkat lunak untuk pemisahan secara jelas diantara sub sistem yang berbeda.

Dengan demikian sebuah sistem akan bisa dimodifikasi secara mudah karena hanya dibutuhkan interface antar class.

5. *Encapsulation*

   Encapsulation adalah proses menyembunyikan detil implementasi sebuah objek. Satu-satunya jalan untuk mengakses data objek tersebut adalah melalui interface. Interface melindungi internal state sebuah objek dari "campur tangan" pihak luar. Oleh karena itu objek sering digambarkan sebagai kotak hitam (black box) yang menerima dan mengirim pesan-pesan (messages). Dalam object-oriented programming kotak hitam tersebut berisi kode (himpunan intruksi dengan bahasa yang dipahami komputer) dan data (informasi dimana intruksi tersebut beroperasi dengannya). Dalam object-oriented programming, kode dan data disatukan dalam sebuah "benda" yang tersembunyi isinya, yaitu objek. Pengguna objek tidak perlu mengetahui isi dalam kotak tersebut. Untuk dapat berkomunikasi dengan objek, diperlukan pesan (messages). Secara formal kita definisikan message sebagai permintaan untuk objek penerima (receiver object) untuk membawa metode yang ditunjukkan atau perilaku dan mengembalikan result dari aksi tersebut kepada objek pengirim (sender object). Sebagai contoh, satu object orang mengirim object bola lampu sebuah pesan (message) untuk menyala (melalui saklar). Objek bola lampu menyalakan dirinya dan menunjukkan kepada objek orang tersebut bahwa state barunya adalah menyala.

6. *Association dan Aggregation*

   Association (asosiasi) adalah hubungan antar objek yang saling membutuhkan. Sedangkan, *aggregation* (agregasi) adalah bentuk khusus dari asosiasi yang menggambarkan seluruh bagian suatu objek merupakan bagian dari objek lainnya. Sebagai contoh, objek tanggal dapat disusun dari objek hari, objek bulan, dan objek tahun. Satu alasan mengapa object-oriented menguntungkan bagi programmer adalah karena programmer dapat mendesain program dalam bentuk objek-objek dan hubungan antar objek tersebut untuk kemudian dimodelkan dalam sistem nyata. Keuntungan yang lain adalah proses pembuatan software dapat dilakukan dengan lebih cepat karena software dibangun dari object-object standar, dapat menggunakan model yang ada, dan dapat membuat model dengan cepat melalui metodologi. Kualitas yang tinggi dari software dapat dicapai karena adanya tested components. Lebih mudah dalam maintenance karena perbaikan kode hanya diperlukan pada satu tempat (bukan diurut dari awal). Mudah dalam membangun sistem yang besar karena subsistem dapat dibuat dan diuji secara terpisah. Mengubah sistem yang sudah ada tidak memerlukan membangun ulang keseluruhan sistem.

2.7.8 Alat Bantu UML

Sebagaimana yang telah dijelaskan diatas tentang UML, bahwa untuk mendapatkan banyak pandangan terhadap sistem informasi yang akan dibangun, UML menyediakan beberapa alat bantu diagram visual yang menunjukan berbagai

aspek dalam sistem. Ada beberapa alat bantu diagram yang disediakan dalam UML antara lain :

1. *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/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 *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

2. *Class Diagram*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/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 siapa 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*.

Hubungan Antar *Class* adalah sebagai berikut :

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.

3. *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 vertical. *Activity diagram* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

4. *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 atar 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.

5. *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*, di mana *message* dari level tertinggi memiliki nomor 1. Messages dari level yang sama memiliki prefiks yang sama.

6. *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.

7. *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 apa), 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.