BAB II

TINJAUAN PUSTAKA

II.1. Pengertian Sistem

Menurut Riyanto dkk (2009: 22) dengan berbagai pendekatan, beragam pula istilah "sistem" didefenisikan. Sistem adalah suatu pengorganisasian yang saling berinteraksi, saling bergantung dan terintegrasi dalam kesatuan variable atau komponen. Terdapat dua kelompok atau elemennya. Pendekatan sistem yang lebih menekankan pada prosedur mendefenisikan sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkelompok dan bekerjasama untuk melakukan kegiatan pencapaian tertentu. Makna dari prosedur sendiri, yaitu urutan yang tepat dari tahapan-tahapan instruksi.; Sedangkan pendekatan yang menekankan pada komponen mendefenisikan sistem sebagai kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu. "Serangkaian atau tatanan elemen-elemen yang diatur untuk mencapai tujuan yang ditentukan sebelumnya melalui pemrosesan informasi".

II.1.1. Data dan Informasi

Menurut Riyanto, dkk (2009: 24). Data merupakan bentuk yang masih mentah yang belum bercerita banyak, sehingga perlu diolah lebih lanjut. Data diolah melalui model tertentu menjadi informasi yang dapat dimanfaatkan oleh penerima dalam membuat keputusan dan melakukan tindakan, yang berarti melakukan suatu tindakan lain yang akan membuat sejumlah data kembali. Data

yang disimpan ini nantinya kembali untuk diolah kembali menjadi informasi. Data tersebut akan ditangkap sebagai input, diproses kembali lewat suatu model perlu diolah melalui sebuah siklus. Siklus ini disebut siklus pengolahan data (*data processing life cycle*).

II.1.2. Sistem Informasi

Menurut Tata Sutabri (2012: 58). Tujuan sistem informasi adalah memenuhi kebutuhan informasi semua manajer dalam perusahaan atau dalam sub-unit organisasi perusahaan. Sistem informasi menyediakan informasi bagi pemakai dalam bentuk laporan dan output dari berbagai simulasi model matematikawan, dimana proses manajemen didefenisikan sebagai aktivitas-aktivitas.

1. Pemrosesan, formulasi terinci untuk mencapai suatu tujuan akhir tertentu adalah aktivitas manajemen yang disebut perencanaan.
2. Pengendalian, perencanaan hanyalah setengah dari pertempuran.

II.2. Sistem Informasi Geografis

Menurut Riyanto, dkk (2009: 35-36). Sistem informasi Geografis (SIG) adalah sistem informasi khusus yang mengelola data yang memiliki informasi spasial (bereferensi keruangan), atau dalam arti sempit, adalah sistem komputer yang memiliki kemampuan untuk membangun, menyimpan, mengelola dan menampilkan informasi bereferensi geografis. Beberapa defenisi dari SIG adalah sebagai berikut :

1. Aronoff menyatakan bahywa SIG sebagai suatu sistem berbasiskan komputer yang digunakan menyimpan dan memanipulasi informasi-informasi geografis, SIG dirancang untuk mengumpulkan, menyimpan, dan menganalisis objek-objek dan fenomena dimana lokasi geografi merupakan karakteristik yang penting atau kritis untuk dianalisis.
2. Subaryono menyatakan bahwa SIG sebagai suatu himpunan terpadu dari *hardware*, *software*, data, dan *liveware* (orang-orang yang bertanggung jawab dalam men-*desain*, mengimplementasikan, dan menggunakan SIG).
3. ESRI (*Environment System Research Institute*) menyatakan bahwa SIG adalah kumpulan yang terorganisir dari perangkat keras komputer, perangkat lunak, data geografis, dan personil yang dirancang secara efisien untuk memperoleh, menyimpan, meng-*update*, memanipulasi, menganalisis, dan menampilkan semua bentuk informasi yang bereferensi geografi.

II.2.1. Karakteristik Sistem Informasi Geografis

1. Merupakan suatu sistem hasil pengembangan perangkat keras dan perangkat lunak untuk tujuan pemetaan, sehingga fakta wilayah dapat disajikan dalam satu sistem berbasis komputer.
2. Melibatkan ahli geografi, informatika dan komputer, serta aplikasi terkait.
3. Masalah dalam pengembangan meliputi cakupan, kualitas dan standar data, struktur, model dan visualisasi data, koordinasi kelembagaan dan etika, pendidikan, *expert system* dan *decision support system* serta penerapannya.

4. Perbedaannya dengan Sistem Informasi lainnya adalah data dikaitkan dengan letak geografis dan terdiri dari data tekstual maupun grafik.
5. Bukan hanya sekedar merupakan pengubahan peta konvensional (tradisional) ke bentuk peta *digital* untuk kemudian disajikan (dicetak/diperbanyak) kembali.
6. Mampu mengumpulkan, menyimpan, mentransformasikan, menampilkan, memanipulasi, memadukan dan menganalisis data spasial dari fenomena geografis suatu wilayah.
7. Mampu menyimpan data dasar yang dibutuhkan untuk penyelesaian suatu masalah. Contoh: Penyelesaian masalah perubahan iklim memerlukan informasi dasar seperti curah hujan, suhu, angina, kondisi awan. Data dasar biasanya dikumpulkan secara berkala dalam jangka yang cukup panjang.

II.2.2. Jenis Data Sistem Informasio Geografis

1. Data Spasial

Data spasial adalah data yang bereferensi geografis atas representasi obyek di bumi. Data spasial pada umumnya berdasarkan peta yang berisikan interprestasi dan proyeksi seluruh fenomena yang berada di bumi. Fenomena tersebut berupa fenomena alamiah dan buatan manusia. Pada awalnya, semua data dan informasi yang ada di peta merupakan representasi dari obyek di muka bumi. Sesuai dfengan perkembangan, peta tidak hanya merepresentasikan obyek-obyek yang ada di muka bumi, tetapi berkembang menjadi representasi obyek diatas muka bumi (diudara) dan dibawah permukaan bumi.

2. Data Vektor

Model data vector adalah yang dapat menampilkan, menempatkan, dan menyimpan data spasial dengan menggunakan titik-titik, garis atau kirva dan *polygon* beserta atribut-atributnya (Prahasta, 2001). Bentuk-bentuk dasar representasi data spasial ini, di dalam sistem model data vector, didefenisikan oleh sistem koordinat kartesian dua dimensi (x, y).

3. Data Raster

Obyek di permukaan bumi disajikan sebagai elemen matriks atau sel-sel *grid* yang homogen. Model data Raster menampilkan, menempatkan dan menyimpan data spasial dengan menggunakan struktur matriks atau piksel-piksel yang membentuk grid (Prahasta, 2001). Tingkat ketelitian model data raster sangat bergantung pada resolusi atau ukurannya terhadap obyek di permukaa bumi.

II.3. Adobe Flash CS 6

Adobe Flash merupakan *authoring tool* yang memudahkan kita untuk mengatur dan mengolah aset. Pada dasarnya kita memanfaatkan Adobe Flash dan Flare3D saja untuk membuat sebuah game atau aplikasi berbasis 3 dimensi. (wandah wibawanto, 2013:29).

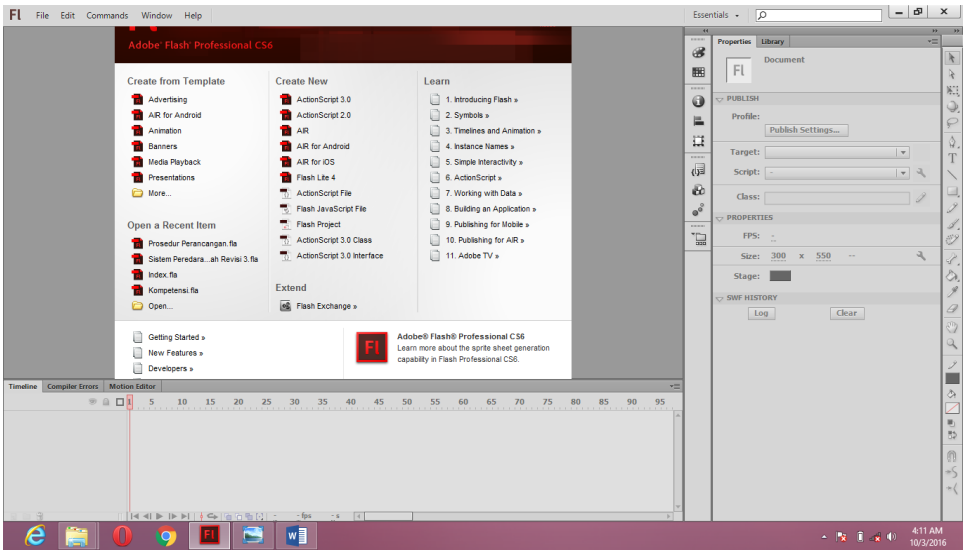
Adobe Flash merupakan software multifungsi yang mempermudah pembuatan animasi, web, game, dan aplikasi multimedia lainnya. Versi terbaru dari Flash yang dimulai dari versi CS5.5 Profesional hingga yang terbaru saat ini

dilengkapi dengan AIR for Android extension. Tidak ada instalasi tambahan yang diperlukan. (Wahana Komputer, 2014: 3, 4).

Adobe Flash CS6 merupakan versi terbaru dari versi sebelumnya, Adobe Flash CS5. Program ini memiliki banyak fungsi, seperti pembuatan animasi objek, membuat presentasi, animasi iklan, game, pendukung animasi halaman web, hingga dapat digunakan untuk pembuatan film animasi. (Wahana Komputer, 2012: 2).

II.3.1. Area kerja macromedia Flash Player

Langkah untuk menjalankan program Adobe Flash CS6, tekan tombol **Start ► All Programs ► Adobe ► Adobe Flash CS6** sehingga tampil **Welcome Screen** seperti tanpak pada gambar berikut, (Madcoms Madium; 2012: 4).



Gambar II.1. Tampilan layar pertama program Adobe Flash Profesional
Sumber : (Madcoms Madium ; 2012 : 4).

Welcome Screen menampilkan empat pilihan perintah untuk memulai Adobe Flash CS6, yaitu:

1. ***Create from Template,*** berguna untuk membuka lembar kerja dengan template yang tersedia dalam program Adobe Flash CS6.
2. ***Open a Recent Item,*** berguna untuk membuka kembali file yang pernah Anda simpan atau pernah Anda buka sebelumnya.
3. ***Create New,*** berguna untuk membuka lembar kerja baru dengan beberapa pilihan script yang tersedia.
4. ***Learn,*** berguna untuk membuka jendela Help yang berguna untuk mempelajari suatu perintah, (Madcoms Madium; 2012: 4-5).

Jika Anda tidak ingin menampilkan jendela Welcome Screen lagi saat membuka program, aktifkan kotak periksa ***Don't Show again*** yang terdapat pada sisi bawah dari jendela Welcome Screen.

II.3.2. Panel Tools

Panel Tools merupakan sebuah panel yang menampung tombol-tombol berguna untuk membuat suatu desain animasi mulai dari tombol seleksi, pen, pensil, 3D Rotate, dan lain-lain. Dalam daftar Tabel II.1. berikut adalah simbol dan nama-nama tombol Toolbox:

Tabel II.1. Fungsi Tombol Panel Tools

Nama Tombol	Fungsi
Selection Tool (V)	Untuk menyeleksi objek.
Subselection Tool (A)	Untuk menyeleksi bagian objek untuk proses editing.
Free Transform	Untuk mengubah bentuk objek secara bebas.

Tool (Q)	
GradientTransform Tool(F)	Untuk mengubah transformasi warna gradasi sebuah objek.
3D Rotation Tool (W)	Untuk melakukan transformasi bentuk dan posisi 3D pada objek berdasarkan sumbu X, Y dan Z.
3D Translation Tool (G)	Untuk melakukan transformasi bentuk dan posisi 3D pada simbol movie clip dengan acuan tiga sumbu X, Y dan Z.
Losso Tool (L)	Untuk menyeleksi objek dengan pola seleksi bebas.
Pen Tool (P)	Untuk menggambar objek
Add Anchor Point Tool (=)	Untuk menambah titik anchor pada sebuah path.
Delete Anchor Point Tool (-)	Untuk menghapus titik Anchor.
Convert Anchor Point Tool (C)	Untuk mengubah sudut lancip dari sebuah path menjadi sudut lengkung.
Text Tool (T)	Untuk mengetik text dan paragraf.
Line Tool (N)	Untuk menggambar objek garis lurus.
Rectangle Tool (R)	Untuk menggambar objek kotak.
Oval Tool (O)	Untuk menggambar objek oval atau lingkaran.
RectanglePrimitive Tool (R)	Untuk menggambar objek kotak dengan sudut dapat di lengkungkan.
Oval Primitive Tool (O)	Unruk menggambar objek lingkaran dengan berbagai variasi.
PolyStar Tool	Untuk menggambar objek poligon dan bintang.
Pencil Tool (Y)	Untuk menggambar dengan bentuk goresan pensil.
Brush Tool (B)	Untuk menggambar dengan bentuk polesan kuas.
Spray Brush Tool (B)	Untuk menggambar dengan spary, yaitu menyemprotkan warna atau simbol.
Deco Tool (U)	Untuk menggambar corak dekorasi dengan menggunakan simbol graphic.
Bone Tool (X)	Membuat animasi pertulangan dengan menggunakan titik sindi pada objek.
Bind Tool (Z)	Melakukan pengeditan dan modifikasi titiksendi dari piranti Bone Tool.
Ink Bottle Tool (S)	Untuk memberi warna dan bentuk garis outline pada sebuah objek.

Paint Bucket Tool (K)	Untuk memberi warna bidang objek.
Eyedropper Tool (I)	Untuk mengambil sample warna dari sebuah objek.
Erasser Tool (E)	Untuk menghapus bidang objek.
Hand Tool (H)	Untuk menggeser area lembar kerja atau stage.
Zoom Tool (M,Z)	Untuk memperbesar atau memperkecil tampilan lembar kerja atau stage.
Stroke Color	Untuk menentukan warna garis.
Fill Color	Untuk menentukan warna bidang objek.
Black and White	Untuk mengubah warna garis dan bidang menjadi hitam dan putih.
Swap Colors	Untuk membalik warna antara warna garis dan warna bidang objek.
No Color	Unruk menghapus warna garis atau warna bidang objek.
Snap to Objects	Untuk mengaktiskan atau mematikan fungsi Snap to Objects.

Sumber : (Madcoms Madium ; 2012 : 6-9).

II.3.3. Timeline

Timeline berguna untuk menentukan durasi animasi, jumlah *layer*, *frame*, menempatkan *script* dan beberapa keperluan animasi lainnya. Semua bentuk animasi yang dibuat akan diatur dan ditempatkan pada layer dalam timeline. (Madcoms Madium ; 2012 : 6-9).

Tabel II.2. Keterangan Tampilan Timeline

Abjad	Nama	Keterangan
A	*Layer*	Layar kerja yang menampung objek yang akan dianimasika di dalam Timeline.
B	*Timeline*	Tabulasi dari lembar kerja atau Stage yang sedang dikerjakan.
C	*Show/Hide All Layers*	Untuk menyembunyikan atau menampilkan semua isi layer.

D	*Lock/Unlock All Layers*	Untuk mengunci atau melepas kunci objek dari semua layer.
E	*Show All Layer as outlines*	Untuk menampilkan objek pada semua layer dalam bentuk outline.
F	*Playhead*	Jarum untuk membaca Frame pada saat animasi dijalankan.
G	*Blang Keyframe*	Sebuah simbol lingkaran kosong yang menampung suatu objek.
H	*Frame*	Suatu bagian dari layer yang digunakan untuk mengatur pembuatan animasi.
I	*Menu*	Untuk mengatur tampilan Frame.
J	*New Layer*	Untuk menambahkan layer baru.
K	*New Folder*	Untuk menambahkan folder baru.
L	*Delete*	Untuk menghapus layer
M	*Simbol Pensil*	Menunjukkan bahwa layer dalam kondisi terpilih atau aktif.
N	*Titik Show or Hide*	Klik untuk menampilkan atau menyembunyikan layer aktif.
O	*Titik Kunci*	Klik untuk mengunci atau melepaskan kunci layer yang aktif.
P	*Kotak Outline*	Klik untuk menampilkan objek dalam layer aktif menjadi bentuk outline.
Q	*Onion skinning button*	Untuk mengatur tampilan animasi didalam stage.
R	*Frame Rate*	Untuk mengatur kecepatan gerak animasi dalam tiap detiknya.
S	*Elapsed Time*	Menunjukkan durasi atau lamanya animasi.
T	*Scrollbar*	Menggulung jendela Timeline secara vertikal dan horisontal.
U	*Current Frame*	Menunjukkan posis Frame aktif.

Sumber : (Madcoms Madium; 2012: 9-10).

II.3.4. Panel *Properties*

Panel *properties* menampilkan perintah dari sebuah tombol yang terpilih sehingga Anda dapat melakukan modifikasi dan memaksimalkan fungsi piranti tersebut.

Melalui panel p*roperties* Anda juga dapat menentukan pengaturan *publish* mulai dari membuka kontak dialog *publish setting*, menentukan versi program Flash ketika menjalankan animasi serta mengatur Stage. (Madcoms Madium ; 2012 : 13).

II.3.5. Stage

Stage adalah lembar kerja yang di gunakan untuk membuat atau mendesain objek yang akan dianimasikan. Objek yang dibuat dalam lembar kerja dapat berupa objek Vektor, Movie clip, Text, Button, dan lain-lain, (Madcoms Madium; 2012: 12).

Tabel II.3. Keterangan Tampilan Stage

Abjad	Keterangan
A	**Stage**, lembar kerja untuk menyusun objek yang akan dianimasikan.
B	**Scene**, menunjukkan nama scene yang aktif.
C	Panah yang digunakan untuk berpindah dari lembar kerja simbol ke lember kerja utama.
D	**Edit Scene**, untuk memilih nama scane yang akan diedit.
E	**Edit Symbols**, untuk memilih nama simbol yang akan diedit.
F	**Zoom**, untuk mengatur besarnya tampilan stage atau lembar kerja.
G	**Scrolber**, untuk menggulung lembar kerja secara horisontal dan vertikal.

Sumber : (Madcoms Madium; 2012: 12).

II.4. Aplikasi *Mobile*

Menurut Wikipedia, pengertian aplikasi adalah program yang digunakan orang untuk melakukan sesuatu pada sistem komputer. *Mobile* dapat diartikan

sebagai perpindahan yang mudah dari satu tempat ke tempat yang lain, misalnya telepon *mobile* berarti bahwa terminal telepon yang dapat berpindah dengan mudah dari satu tempat ke tempat lain tanpa terjadi pemutusan atau terputusnya komunikasi. Sistem aplikasi *mobile* merupakan aplikasi yang dapat digunakan walaupun pengguna berpindah dari satu tempat ke tempat lain tanpa terjadi pemutusan atau terputusnya komunikasi. Aplikasi ini dapat diakses melalui perangkat nirkabel seperti pager, seperti telepon seluler dan PDA.

Adobe Air berjalan di atas *platform Flash* dan memungkinkan penggunaan fungsi-fungsi dan *tools* yang dimiliki oleh *Adobe Flash* ke dalam pengembangan aplikasi berbasis *Android*. *Adobe Flash* merupakan software multifungsi yang mempermudah pembuatan animasi, *web, game,* dan aplikasi multimedia lainnya.

ActionScript adalah bahasa pemrogarman untuk *Adobe Flash Player* dan *Adobe AIR Environment*. *ActionScript* dijalankan oleh *ActionScript Virtual Machine* yang merupakan bagian dari *Flash player* dan *AIR*. Koding *ActionScript* biasanya dikompilasi ke dalam format *bytecode* (bahasa pemograman yang ditulis dan dipahami oleh komputer) oleh *compiler*, seperti yang dibangun kedalam *Adobe Flash CS6 Professional* atau *Adobe Flex Builder* dan yang tersedia di dalam *Adobe Flex SDK* dan *Flex Data Services*. *Bytecode* ini tertanam dalam file SWF, yang dijalankan oleh *Flash player* maupun *AIR*. (Wahana Komputer, 2014: 3).

Android adalah system operasi berbasis Linux yang digunakan untuk telepon seluler (mobile) seperti telepon pintar (smartphone) dan computer tablet (PDA). Android menyediakan platform terbuka bagi para pengembang untuk

menciptakan aplikasi mereka sendiri yang digunakan oleh bermacam peranti bergerak. Android kini telah menjelma menjadi system operasi mobile terpopuler di dunia. Perkembangan Android kini telah menjelma menjadi system operasi mobile terpopuler di dunia. Perkembangan Android tidak lepas dari peran sang raksasa Google. Android pada mulanya didirikan oleh Andy Rubin, Rich Miner, Nick Sears dan Chris White pada tahun 2003. (Yosef Murya, 2014: 3).

Ponsel pertama dengan sistem operasi Google Android akhirnya dirilis pada tahun 2008. Ponsel buatan produk HTC yang dinamakan T-Mobile G1 itu diluncurkan di New York dan kala itu digadang-gadang bakal menggoyang dominasi smartphone Apple iPhone ataupun BlackBerry. (Yosef Murya, 2014: 6).

II.5. Arsitektur GIS

Menurut Riyanto, dkk (2009: 38) Beberapa subsistem dalam sistem informasi geografis antara lain adalah:

1. *Input*

   Pada tahap *input* (pemasukan data) yang dilakukan adalah mengumpulkan dan mempersiapkan data spasial dan data atribut dari berbagai sumber data. Data yang digunakan harus dikonversikan menjadi format *digital* yang sesuai. Proses konversi yang dilakukan dikenal dengan proses dijitalisasi (*digitizing*).

2. Manipulasi

   Manipulasi data merupakan proses *editing* terhadap data yang telah masuk, hal ini dilakukan untuk menyesuaikan tipe jenis data agar sesuai dengan sistem

yang akan dibuat, seperti: penyamaan skala, pengubahan sistem, proyeksi, generalisasi dan sebagainya.

3. Analisis

   Terdapat dua jenis fungsi analisis dalam SIG, yaitu fungsi analisis spasial dan analisis atribut. Fungsi analisis spasial adalah operasi yang dilakukan pada data spasial. Sedangkan, fungsi analisis atribut adalah fungsi pengolahan data atribut, yaitu data yang tidak berhubungan dengan ruang

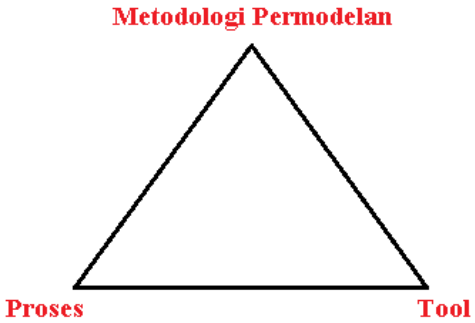
4. Visualisasi (*Data Output*)

   Penyajian hasil berupa informasi baru atau *database* yang ada baik dalam bentuk *softcopy* maupun dalam bentuk *hardcopy* seperti dalam bentuk peta : peta (atribut peta dan atribut data), tabel, grafik, dan lain-lain.

II.6. UML (*Unified Modelling Language*)

Pemodelan (*modeling*) adalah proses merancang piranti lunak sebelum melakukan pengkodean (*coding*). Model piranti lunak dapat dianalogikan seperti pembuatan blueprint pada pembangunan gedung. Membuat model dari sebuah sistem yang kompleks sangatlah penting karena kita tidak dapat memahami sistem semacam itu secara menyeluruh. Semakin komplek sebuah sistem, semakin penting pula penggunaan teknik pemodelan yang baik.Dengan menggunakan model, diharapkan pengembangan piranti lunak dapat memenuhi semua kebutuhan pengguna dengan lengkap dan tepat, termasuk faktor-faktor seperti *scalability*, *robustness*, *security*, dan sebagainya. Kesuksesan suatu pemodelan piranti lunak ditentukan oleh tiga unsur, yang kemudian terkenal dengan sebuan

segitiga sukses (*the triangle for success*). Ketiga unsur tersebut adalah metode pemodelan (*notation*), proses (*process*) dan *tool* yang digunakan. Memahami notasi pemodelan tanpa mengetahui cara pemakaian yang sebenarnya (proses) akan membuat proyek gagal. Dan pemahaman terhadap metode pemodelan dan prosesdisempurnakan dengan penggunaan tool yang tepat.

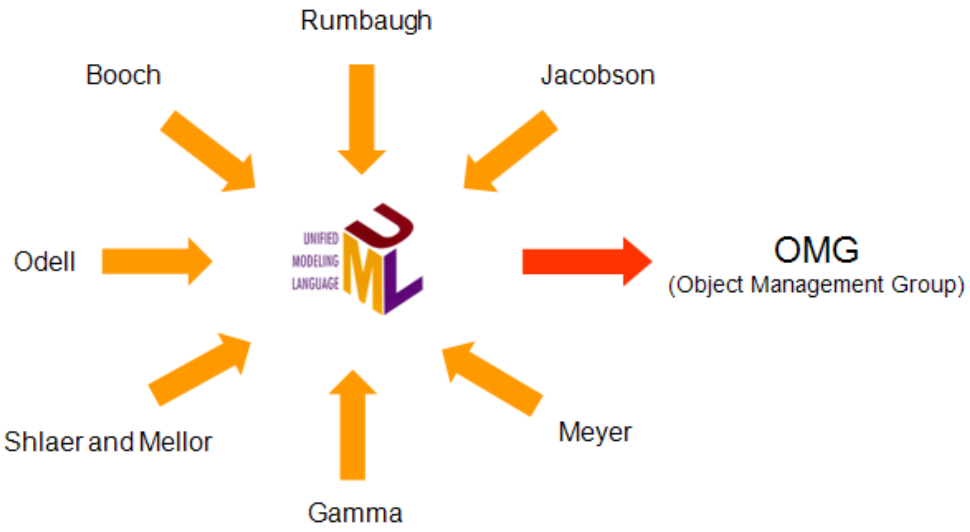


Gambar II.2.*The Triangle For Success*
Sumber :(*Yuni Sugiarti ; 2013 : 33*)

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.Seperti bahasa-bahasa lainnya, UML

mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*). Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: *metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock*, dsb. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan group/perusahaan lain yang menggunakan metodologi yang berlainan.Dimulai pada bulan Oktober 1994 *Booch, Rumbaugh dan Jacobson*, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakanmempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease draft pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh Object Management Group (OMG – http://www.omg.org). Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003.Booch, Rumbaugh dan Jacobson menyusun tiga

buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek.



Gambar II.3. Metodelogi Pemodelan Berorientasi Objek
Sumber : (*Yuni Sugiarti; 2013 : 33*)

Dalam hal ini, pemodelan perangkat lunak bekerja dengan cara yang cukup serupa layaknya seorang arsitek atau insinyur teknik sipil yang akan membuat sebuah bangunan/dedung berskala besar. Saat seorang arsitek atau insinyur teknik sipil akan membuat sebuah bangunan gedung berskala besar, ia biasanya membuat denah-denah atau maket-maket yang menggambarkan bentuk jadi dari bangunan/gedung.

Dari berbagai penjelasan rumit yang terdapat di dokumen dan buku-buku UML. Sebenarnya konsepsi dasar UML bisa kita rangkumkan dalam gambar dibawah.

Tabel II.4 Konsep Dasar UML

Major Area	***View***	***Diagram***	***Main Concept***
Structural	*Static View*	*Class Diagram*	*Class, Association, Generalization, Dependency, Realization, Interface*
	Use Case View	*Use Case Diagram*	*Use Case, Actor, Association, Extend, Include, Use Case Generalization*
	Implementation view *Deployment view*	*Component Diagram*	*Component, Interface, Dependency, Location*
		Deployment Diagram	*Node, Component, Depedency, Location*
	Interaction View	*Sequence Diagram*	*Interaction, Object, Message, Activation*
		Class Diagram	*Collaboration, interaction, Collaboration Role, Message*
Model Management	*Model Management View*	*Class Diagram*	*Package, Subsystem, Model*
Extensibility	*All*	*All*	*Constraint, Stereotype, Tagged Values*

Sumber :(*Yuni Sugiarti ; 2013 : 35*)

Seperti juga tercantum pada gambar diatas UML mendefinisikan diagram-diagram sebagai berikut:

1. *Use case diagram*
2. *Class diagram*
3. *Statechart diagram*
4. *Activity diagram*
5. *Sequence diagram*
6. *Collaboration diagram*

7. *Component diagram*
8. *Deployment diagram*

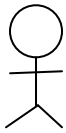
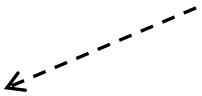
Dalam pembuatan skripsi ini penulis menggunakan diagram Use Case yang terdapat di dalam UML. Adapun maksud dari Use Case Diagram diterangkan dibawah ini.

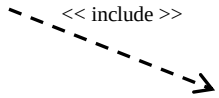
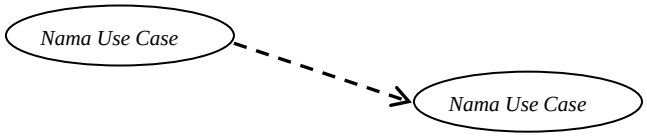
II.6.1. *Use Case Diagram*

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. Secaraumum 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-*extenduse case* lain dengan *behaviour*-nya sendiri.

Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

Tabel II.5. Simbol *Use Case*

Simbol	Deskripsi
Use Case  	Fungsional yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit dan aktor, biasanya menggunakan kata kerja di awal frase nama *use case*
Aktor  *Nama Aktor*	Orang atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat sistem itu sendiri. Jadi walaupun simbol aktor adalah gambar orang, tapi aktor belum tentu merupakan orang. Biasanya menggunakan kata benda di awal frase nama aktor
Asosiasi / Association	Komunikasi antar aktor dan *use case* yang berpartisipasi *use case* atau *use case* berinteraksi dengan aktor.
Extend  << extend >>	Relasi *usecase* tambahan ke sebuah *usecase* dimana *use case* yang ditambahkan dapat berdiri sendiri walupun tanpa *usecase* tambahan itu, mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *usecase* tambahan memiliki nama depan yang sama dengan *usecase* yang ditambahkan, arah panah menunjukan pada *usecase* yang dituju. Contoh :

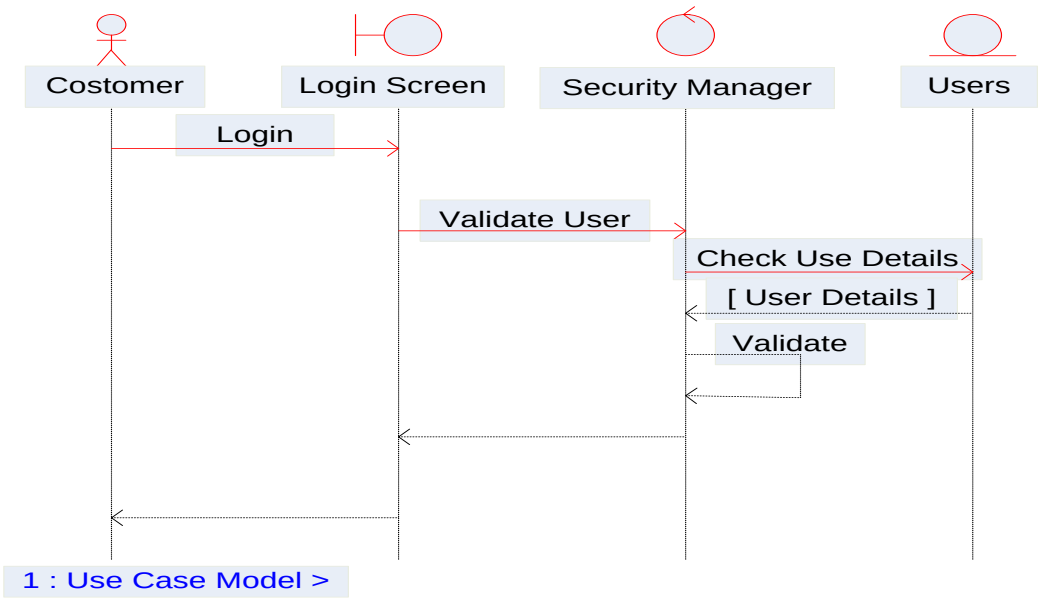
	
Include 	Relasi *usecase* tambahan ke sebuah *usecase* dimana *usecase* yang ditambahkan memerlukan *usecase* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *usecase* ini. ada dua sudut pandang yang cukup besar mengenai *include* di *usecase*, *include* berarti *usecase* yang ditambahkan akan selalu dipanggil saat *usecase* tambahan dijalankan, contoh; 

Sumber :(*Yuni Sugiarti ; 2013 : 42*)

II.6.2. *Sequence Diagram*

Diagram *Sequence* menggambarkan kelakuan/prilaku objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram *sequence*maka harus diketahui objek-objek yang terlibat dalam sebuah *usecase* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya diagram *sequence* yang harus digambar adalah sebanyak pendefinisian *usecase* yang memiliki proses sendiri atau yang penting semua *usecase* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram *sequence* sehingga semakin banyak *usecase* yang didefinisikan maka diagram *sequence* yang harus dibuat juga semakin banyak.



Gambar II.4. Contoh *Sequence* Diagram
Sumber : (*Yuni Sugiarti ; 2013 : 63*)

II.6.3.*Activity Diagram*

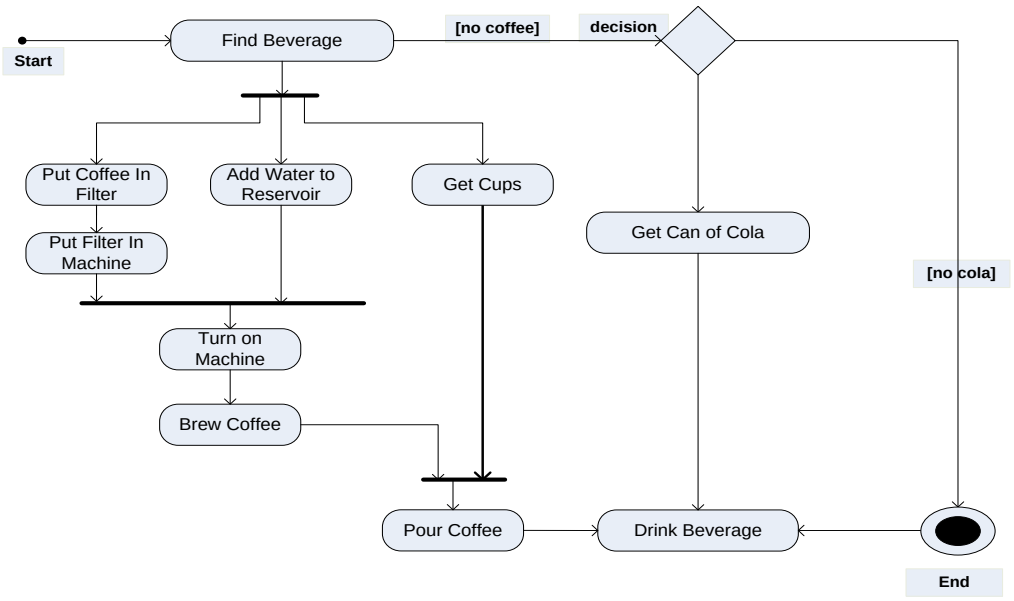
Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

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

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

Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal.

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



Gambar II.12. Activity Diagram
Sumber :(*Yuni Sugiarti ; 2013 : 76*)