BAB II
LANDASAN TEORI

2.1 Rancang Bangun

Menurut Jogiyanto (2005), rancang bangun (desain) adalah tahap dari setelah analisis dari siklus pengembangan sistem yang merupakan pendefinisian dari kebutuhan-kebutuhan fungsional, serta menggambarkan bagaimana suatu sistem dibentuk yang dapat berupa penggambaran, perencanaan dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah ke dalam satu kesatuan yang utuh dan berfungsi, termasuk menyangkut mengkonfigurasikan dari komponen-komponen perangkat keras dan perangkat lunak dari suatu sistem.

2.2 Aplikasi

Aplikasi adalah suatu program komputer yang dibuat untuk mengerjakan dan melaksanakan tugas khusus dari pengguna. Aplikasi merupakan rangkaian kegiatan atau perintah yang dieksekusi oleh komputer. Program merupakan kumpulan *instruction set* yang akan dijalankan oleh pemroses, yaitu berupa *software*. Bagaimana sebuah sistem komputer berpikir diatur oleh program ini. Program inilah yang mengendalikan semua aktivitas yang ada pada proses. Program berisi konstruksi logika yang dibuat manusia, dan sudah diterjemahkan ke dalam bahasa mesin sesuai dengan format yang ada pada *instruction set*. Program aplikasi merupakan program siap pakai. Program direkam untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain. Aplikasi akan menggunakan *Operating System* (OS) komputer dan aplikasi yang mendukung. (Wardana, 2010).

2.3 Pengertian Media Pembelajaran

Menurut Heinich yang dikutip oleh Azhar Arsyad (2011), media pembelajaran adalah perantara yang membawa pesan atau informasi bertujuan instruksional atau mengandung maksud-maksud pengajaran antara sumber dan penerima.

2.4 Pengertian Lalu Lintas

Lalu lintas didalam Undang-undang No 22 tahun 2009 didefinisikan sebagai gerak Kendaraan dan orang di Ruang Lalu Lintas Jalan, sedang yang dimaksud dengan Ruang Lalu Lintas Jalan adalah prasarana yang diperuntukkan bagi gerak pindah Kendaraan, orang, dan barang yang berupa jalan dan fasilitas pendukung (Umbara, 2009).

2.4.1 Rambu – Rambu Lalu Lintas

Menurut Julianto (2008) Rambu lalu lintas adalah salah satu alat perlengkapan jalan dalam bentuk tertentu yang memuat lambang, huruf,angka, kalimat dan perpaduan di antaranya, yang digunakan untuk memberikanperingatan, larangan, perintah dan petunjuk bagi pemakai jalan. Rambu lalu lintas dibuat untuk menciptakan kelancaran, keteraturan dan keselamatan dalam berkendara. Marka jalan dan rambu – rambu merupakan objek untuk menyampaikan informasi baik itu perintah, larangan, dan petunjuk.

Dalam diktat Rekayasa Lalu Lintas (2008) rambu – rambu lalu lintas mengandung berbagai fungsi yang masing – masing mengandung konsekuensi hukum sebagai berikut :

1) Perintah

Yaitu bentuk pengaturan yang jelas dan tegas tanpa ada interpretasi lain yang wajib dilaksanakan oleh pengguna jalan. Karena sifatnya perintah, maka tidak benar adanya perintah tambahan yang membuka peluang munculnya interpretasi lain. Misalnya : rambu belok kiri yang disertai kalimat belok kiri boleh terus adalah bentuk yang keliru. Penggunaan kata boleh dan terus mengandung makna ganda dan dengan demikian mengurangi makna perintah menjadi makna pilihan. Yang benar adalah belok kiri langsung. Dengan demikian, pelanggar atas perintah ini dapat dikenai sanksi sesuai perundang – undangan yang berlaku.

2) Larangan

Yaitu bentuk larangan yang dengan tegas melarang para pengguna jalan untuk melakukan hal – hal tertentu. Tidak ada pilihan lain kecuali tidak dilakukan.

3) Peringatan

Menunjukan kemungkinan adanya bahaya di jalan yang akan dilalui. Rambu peringatan berbentuk bujur sangkar berwarna dasar kuning dan lambang atau tulisan berwarna hitam.

4) Anjuran

Yaitu bentuk pengaturan yang bersifat mengimbau, boleh dilakukan boleh pula tidak. Pengemudi yang melakukan atau tidak melakukan anjuran tersebut tidak dapat disalahkan dan dikenakan sanksi.

5) Petunjuk

Yaitu memberikan petunjuk mengenai jurusan, keadaan jalan, situasi, kota berikutnya, keberadaan fasilitas dan lain – lain.

Bentuk dan warna yang digunakan pada rambu – rambu lalu lintas digunakan untuk membedakan kategori rambu – rambu yang berbeda namun memberikan kemudahan bagi pengemudi dan membuat pengemudi lebih cepat untuk bereaksi. Berikut ini adalah contoh gambar dan penjelasan rambu – rambu lalu lintas yaitu sebagai berikut :

Gambar 2.1 rambu – rambu lalu lintas

2.5 Aplikasi Android

Menurut M. Hilmi Masrury dan Java Creativity (2015), Android merupakan sistem operasi berbasis *linux* untuk perangkat *mobile*. Android merupakan sistem operasi gratis dan *open source*, jadi Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan suatu aplikasi sendiri yang mampu berjalan di atas peranti *Android*, hal itulah yang menjadikan Android mampu bersaing di tengah keramaian *smartphone blackberry* dan *iphone* yang lebih dahulu meramaikan pasaran.

2.5.1 Versi Android

Keunikan dari sistem operasi Android ini adalah dari tiap nama yang diberikan pada tiap versinya yang di mana nama tersebut menggunakan nama sebuah makanan penutup di sebuah restoran. Tidak hanya itu saja, keunikan dari nama tiap versi Android urut sesuai abjad. Berikut ini perjalanan versi sistem operasi Android yang telah diberikan oleh google. Menurut M. Hilmi Masrury dan Java Creativity (2015).

a. Android Versi Beta

   Android versi beta ini dirilis pada tanggal 5 November 2007, di mana Android pertama kali muncul dan masih belum digunakan oleh orang seperti pada saat ini.

b. Android Versi 1.0

   Android Versi 1.0 ini dirilis pada tanggal 23 November 2008, ponsel pertama yang menggunakan sistem operasi ini yaitu HTC G1 (*Dream*), pada Android 1.0 ada yang memberi pernyataan dengan nama *Angel Gake* dan ada juga yang memberikan pernyataan dengan nama *Appel Pie*.

c. Android Versi 1.1

   Android versi 1.1 ini dirilis pada tanggal 9 Maret 2009. Pada versi ini telah dilakukan perbaikan-perbaikan terhadap sejumlah permasalahan yang ditemukan pada Android versi 1.0 sebelumnya. Begitu juga pada

Android versi 1.1 ini juga ada perbedaan nama versi yaitu : *Battenberg* dan *Banana Bread*.

d. Android Versi 1.5

Android versi 1.5 ini dirilis oleh Google pada tanggal 30 April 2009. Pada versi inilah, Google pertama kali memberikan nama kode atau *code name* pada versi Android, yaitu dengan nama *CUPCAKE* yang artinya KUE MANGKUK.

e. Android Versi 1.6

Android versi 1.6 ini dikeluarkan oleh Google pada tanggal 15 September 2009. Kali ini Google memberikan kode nama *Donut*, yaitu sebuah makanan yang sudah tidak asing lagi di telinga kita dan pada versi ini pula telah hadir Android *market* baru.

Android 1.6 dirilis untuk memperbaiki kesalahan *reboot* pada sistem operasi serta perubahan fitur pada antarmuka kamera dan integrasi pencarian yang lebih baik.

f. Android Versi 2.0 dan 2.1

Android versi 2.0 ini dirilis pada tanggal 26 Oktober 2009 selang waktu tiga bulan Google telah merilis kembali sistem operasi Android dengan versi 2.1 tepatnya pada tanggal 12 Januari 2010, kedua versi tersebut telah diberi nama kode yang sama, yaitu *ECLAIR*, di mana *Eclair* merupakan makanan penutup yang biasanya dihidangkan bersama dengan secangkir kopi hangat pada sebuah restoran.

g. Android Versi 2.2

Android versi 2.2 ini dirilis oleh Google pada tanggal 20 Mei 2010 dengan nama *FROYO*, *code name* tersebut merupakan singkatan dari *Frozen Yoghurt* yang merupakan sebuah makanan beku yang dibuat dari *yoghurt*. Pada Android versi ini sangat banyak sekali penambahan fitur-fitur baru dan perbaikan yang cukup besar pengaruhnya terhadap ponsel Android.

h. Android Versi 2.3

Android versi 2.3 ini dikeluarkan oleh Google pada tanggal 6 Desember 2010. Kali ini Google memberikan *code name GINGERBRAD* yang memiliki arti ROTI JAHE. Pada versi ini, Android secara resmi mendukung beberapa perangkat keras baru, seperti kamera yang lebih dari satu, perangkat *Near Field Communication* (NFC), sensor *gyroscope*, dan sensor barometer.

i. Android Versi 3.0 dan 3.1

Android versi 3.0 diluncurkan Google pada tanggal 22 Februari 2011 dengan memberikan nama kode *HONEYCOMB* yang berarti SARANG MADU tak lama kemudian Google merilis versi terbarunya yaitu 3.1 dengan *code name* yang masih lama. Android versi ini khusus dirancang untuk digunakan pada perangkat *tablet* yang memiliki antara 5 - 10 inch.

j. Android Versi 4.0

Android versi 4.0 dirilis pada tanggal 19 Oktober 2011 yang diberikan nama *ICE CREAM SANDWICH*, versi ini merupakan keluaran terbaru dari Google yang membawa fitur *Honeycomb* untuk *smartphone*.

k. Android Versi 4.1, 4.2, dan 4.3

Dikenalkan pada 27 Juni 2012 pada konferensi Google I/O. Perkembangan banyak terjadi pada versi ini yang membuatnya menjadi versi Android yang tercepat dan terhalus yang pernah ada. *Jelly Bean* dianggap sebagai versi Android yang memiliki kinerja paling cepat dan terhalus. Resmi diperkenalkan di sebuah konferensi Google I/O pada Juni 2012, versi Android yang satu ini mampu memberikan pengalaman luar biasa bagi penggunanya dengan menyajikan tampilan *interface* serta *Google Search* yang dibekali kemampuan baru.

l. Android Versi 4.4

Sebelum dirilis resmi, para pengamat *gadget* memprediksi bahwa untuk versi lanjutan dari *Jelly Bean* akan diberi nama *"Key Lime Pie"*, tapi ternyata rumor tersebut salah kaprah. Penanaman untuk versi ini cukup

mencengangkan karena mengambil nama produk coklat yang memang sudah terkenal sebelumnya atau tergolong komersial yaitu *"Kitkat"*.

m. Android Versi 5.0

Android *Lollipop* ini baru saja dirilis pada tanggal 15 Oktober 2014. Meskipun pada saat itu *Lollipop* baru dalam masa percobaan akan tetapi komentar yang masuk terbilang bagus.

2.5.2 Kelebihan Android

Menurut Ed Burnette (2010) dalam bukunya *Hello Android*, *Operating System* Android memiliki keunggulan antara lain :

a. Sangat terbuka, *development platform* berbasis *Linux* dan *Open source* membuat para pembuat *handset* dapat mengeditnya tanpa membayar *royalty*. *Developers* menyukainnya karena mereka tahu bahwa *platform* ini "memiliki kaki" dan tidak terikat dengan *vendor* manapun yang dapat berada jauh dibawah ataupun diakuisisi.

b. Arsitektur berbasis *component* yang terinspirasi dari kebebasan dunia internet. Bagian dari sebuah aplikasi dapat digunakan kembali dengan cara yang tidak sama seperti apa yang dikembangkan *developer*. Ini akan membuat kreativitas baru di dalam dunia *mobile.*

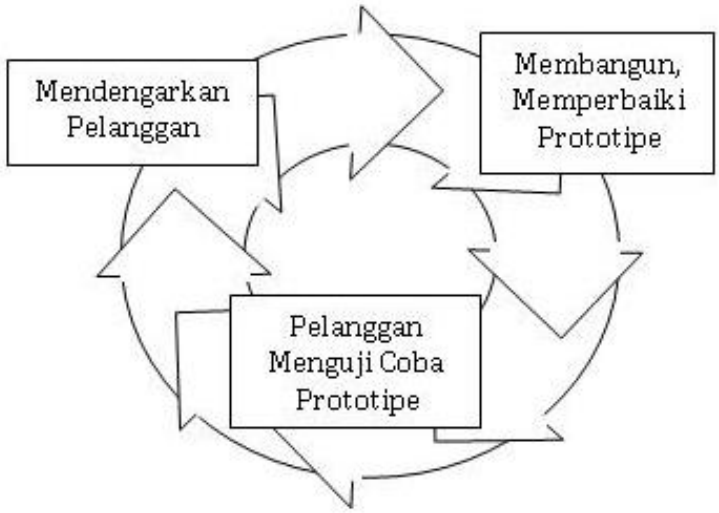
c. Grafis berkualitas tinggi dan suara halus, grafis vektor *antialiased 2D* dan animasi terinspirasi oleh *Flash* yang menyatu dengan *3Deaccelerated OpenGL* grafis untuk memungkinkan jenis baru permainan dan aplikasi bisnis. *Codec* untuk standar industri yang paling umum, audio, dan video format yang dibangun di tepat dalamnya, termasuk *H.264 (AVC)*, *MP3*, dan *AAC*.

d. Memiliki portabilitas di berbagai perangkat keras saat ini dan masa depan. Semua program ditulis di *Java* dan dieksekusi oleh mesin virtual *Android* (*Dalvik Android*), sehingga kode Anda akan portabel di *ARM*, arsitektur *x86*, dan lainnya. Dukungan untuk berbagai masukan termasuk metode seperti *keyboard*, sentuhan, dan *trackball*. Antarmuka pengguna dapat disesuaikan untuk setiap resolusi layar dan orientasi (Ed Burnette, 2010).

2.6 Metode Pengembangan Perangkat Lunak

Menurut Rosa A.S. dan M. Shalahuddin (2013), model prototipe (*prototyping model*) dimulai dari mengumpulkan kebutuhan pelanggan terhadap perangkat lunak yang akan dibuat. Lalu dibuatlah program prototipe agar pelanggan lebih terbayang dengan apa yang sebenernya diinginkan. Program prototipe biasanya merupakan program yang belum jadi. Program ini biasanya menyediakan tampilan dengan simulasi alur perangkat lunak sehingga tampak seperti perangkat lunak yang sudah jadi. Program prototipe ini dievalusai oleh pelanggan atau user sampai ditemukan spesifikasi yang sesuai dengan keinginan pelanggan atau user.

Berikut adalah gambar dari model prototipe :



Gambar 2.2 Sistem Model *Prototyping*

Pendekatan *Prototyping* melewati tiga proses, yaitu pengumpulan kebutuhan, perancangan, dan evaluasi *Prototype*. Proses-proses tersebut dapat dijelaskan sebagai berikut :

1) Pengumpulan kebutuhan : *developer* dan klien bertemu dan menentukan tujuan umum, kebutuhan yang diketahui dan gambaran bagian-bagian yang akan dibutuhkan berikutnya, dimana komunikasi disini terjadi.

2) Perancangan : perancangan dilakukan cepat dan rancangan mewakili semua aspek *software* yang diketahui, dan rancangan ini menjadi dasar pembuatan *prototype*;

3) Evaluasi *Prototype* : pada proses ini adanya penyerahan sistem atau perangkat lunak kepada klien, sehingga klien mengevaluasi *prototype* yang dibuat dan digunakan untuk memperjelas kebutuhan *software*.

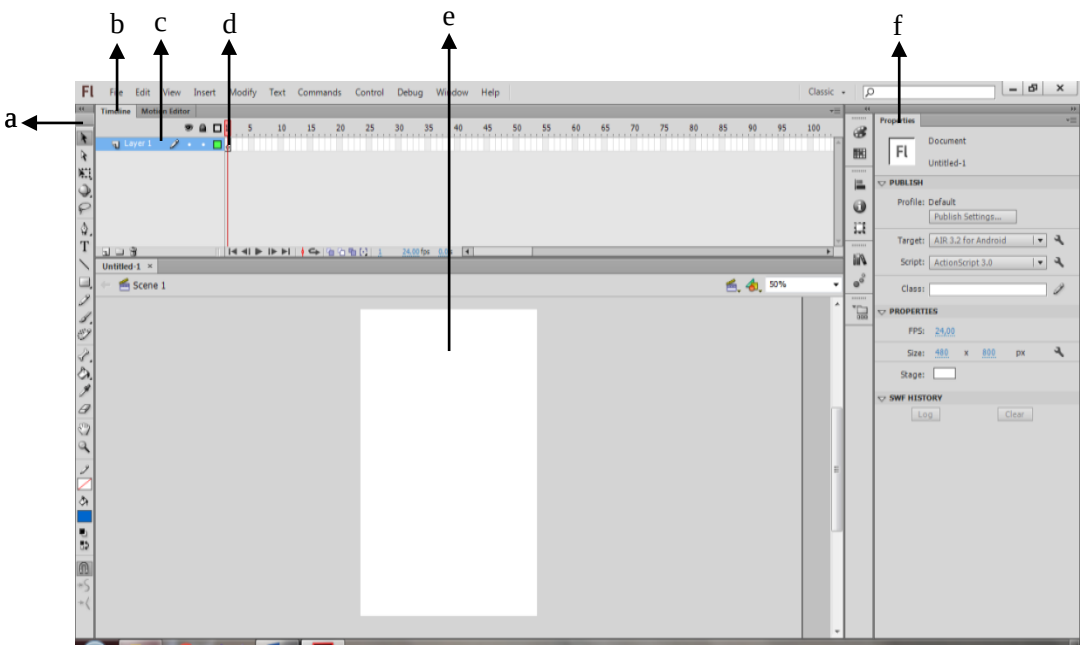
Perulangan ketiga proses ini terus berlangsung hingga semua kebutuhan terpenuhi. *prototype-prototype* dibuat untuk memuaskan kebutuhan klien dan untuk memahami kebutuhan klien lebih baik. *Prototype* yang dibuat dapat dimanfaatkan kembali untuk membangun *software* lebih cepat, namun tidak semua *prototype* bisa dimanfaatkan. Sekalipun *prototype* memudahkan komunikasi antar *developer* dan klien, membuat klien mendapat gambaran awal dari *Prototype*.

2.7 Adobe Flash

Adobe AIR berjalan diatas *platform Flash* dan memungkinkan penggunaan fungsi-fungsi dan *tools* yang dimiliki oleh *Adobe Flash* kedalam pengembangan berbasis *Android. Adobe Flash* merupakan *software* multifungsi yang mempermudah pembuatan animasi, web, game dan aplikasi multimedia lainnya.

2.8 Adobe flash professional CS6

Menurut Irman Maulana (2014), pada dasarnya baik *Flash* versi *Macromedia* maupun versi *Adobe*, area kerjanya terbagi menjadi beberapa bagian, di antaranya menu, *toolbox*, *timeline*, *stage*, dan *panel*.



Gambar 2.3 Tampilan area kerja Adobe Flash CS6

a. *Toolbox*

Di dalam *toolbox* terdapat macam-macam *tool* yang bisa kita gunakan untuk menggambar objek. *Toolbox* ini terdiri dari beberapa bagian yaitu *selection tool, drawing tool, painting tool,* dan *navigation tool.*

b. *Timeline*

Timeline pada *Adobe Flash CS6* secara *default* terletak diatas *stage*. *Timeline* berfungsi untuk mengontrol keseluruhan objek dan animasi yang terdapat pada *stage*.

c. *Layer*

Layer dapat diilustrasikan seperti tumpukan-tumpukan yang berisi objek di dalamnya. Dengan *layer*, kita bisa mengatur *movie* pada *stage* dan menentukan kedalaman atau lapisan suatu objek. Untuk menambahkan *layer* baru, kita dapat memilih menu *Insert* > *Timenline* > *Layer* atau dengan mengklik *icon* yang terdapat pada bagian bawah *timeline*.

Terdapat beberapa mode yang dapat dipilih pada *layer* di ataranya :

- *Guide layer*, berfungsi untuk membuat animasi dengan menggunakan jalur atau *track* berupa garis yang telah dibuat.
- *Motion layer*, digunakan untuk membuat pergerakan animasi *tween*.
- *Masking layer*, digunakan untuk membuat animasi efek *masking*.

d. *Frame*

Frame merupakan bagian dari *layer* yang digunakan untuk mengatur pergerakan animasi. Di dalam *frame* bisa terdiri dari teks, gambar, audio,video, dan kode program *ActionScript*.

Frame terdiri dari beberapa bentuk, di antaranya :

- *Keyframe*, *frame* yang memiliki bentuk bulatan hitam, menandakan bahwa di dalamnya terdapat objek.
- *Blank keyframe*, *frame* berbentuk bulat putih, menandakan bahwa *frame* masih kosong.
- *Action frame*, *frame* dengan bulatan putih dan terdapat huruf 'a', menandakan di dalamnya terdapat kode program *ActionScript*.
- *Frame label*, yaitu *frame* yang diberi nama *label* dan ditandai dengan bendera warna merah.

e. *Stage*

Stage adalah area putih berbentuk kotak yang terletak ditengah area kerja *Flash*. Seperti istilah *stage* pada panggung teater, *stage* di dalam *Flash* berfungsi untuk menampilkan semua objek maupun *movie* yang berjalan diatasnya. Objek yang diletakan pada *stage* bisa berupa teks, gambar, dan video.

f. Panel *Properties*

Panel ini menampung semua properti yang terdapat pada *tool-tool* dalam *Flash*. Untuk mengaktifkannya, pilih menu *Windows* > *Properties*.

g. Panel *Library*

Seperti layaknya perpustakaan, *library* pada *Flash* berfungsi menampung semua simbol yang telah dibuat maupun hasil impor dari file luar seperti

audio, video, gambar, dll. Untuk menampilkan *panel library,* bisa melalui menu *Window* > *Library* atau dengan menekan tombol CTRL+L.

h. Panel *Action*

Panel ini berfungsi sebagai tempat menuliskan *ActionScript*. Untuk mengaktifkannya, pilih menu *Window* > *Action* atau menekan tombol F9.

2.9 ActionScript 3.0

Menurut Irman Maulana (2014), *ActionScript 3.0* atau disingkat AS3 merupakan bahasa pemrograman yang bekerja pada *Adobe Flash*, *Flex*, dan *FlashDevelop*. *ActionScript 3.0* pertama kali dirilis pada tahun 2006 bersamaan dengan diluncurkannya *Flash* versi 9 sekaligus *Flash* pertama yang kini telah diakuisisi oleh *Adobe System Inc.* yaitu *Adobe Flash CS3*.

ActionScript 3.0 adalah bahasa pemrograman yang didasarkan pada *ECMAScript*, yaitu standar bahasa pemrograman yang dikembangkan oleh *ECMA* (*Europan Computer Manufacturers Association)*. Dengan standar ini, AS3 mampu melakukan integrasi data yang cepat dengan berbagai bahasa pemrograman lainnya seperti *JavaScript* dan *XML*.

2.10 Adobe AIR

Menurut Chamber (2008), *Adobe AIR* (*Adobe Integrated Runtime*) adalah sebuah *cross-operation-system runtime* yang di kembangkan oleh *Adobe* untuk mengijinkan para *developers web* untuk memanfaatkan keahlian pengembangan web (seperti *Flash, Flex, HTML, Java-Script,* dan *PDF*) untuk membuat dan menyebarkan *AIR* dan konten ke *desktop*.

Pada dasarnya, *Adobe AIR* menyediakan sebuah *platform* di antara *dekstop* dan *browser*, yang mana menggabungkan jangkauan dan kemudahan terhadap pengembangan suatu model web dengan kegunaan dan kekayaan dari model *desktop*.

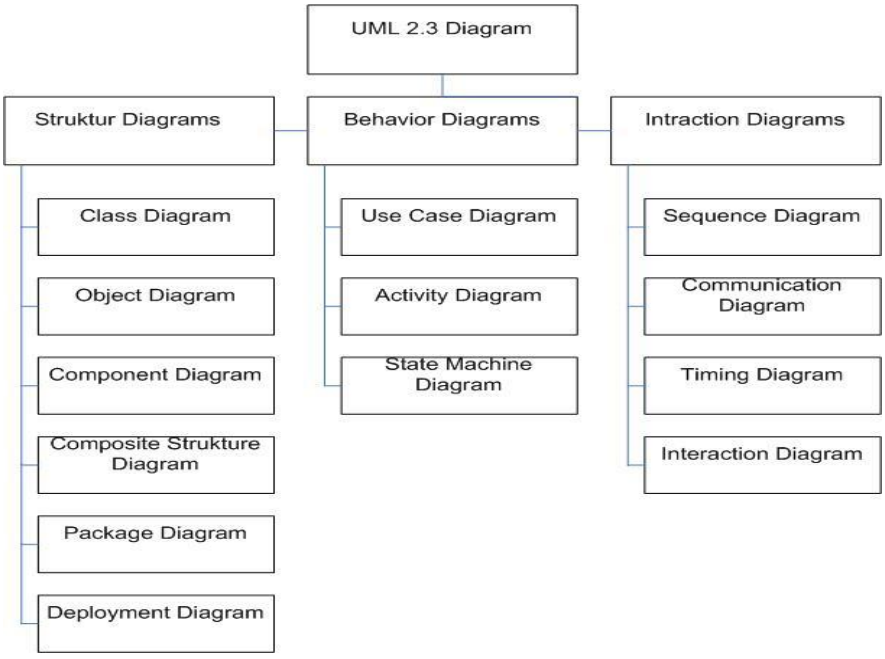
2.11 UML *(Unified Modeling Language)*

Rosa A.S dan M. Shalahuddin (2013), *Unified Modelling Language* atau biasa disingkat UML merupakan bahasa visual untuk pemodelan dan komunikasi

mengenai sebuah sistem dengan menggunakan diagram dan teks- teks pendukung. UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metodologi berorientasi objek. 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.

2.11.1. Diagram-Diagram UML

Rosa A.S dan M. Shalahuddin (2013), pada UML 2.3 tediri dari 13 macam diagram yang dikelompokkan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat Gambar 2.2 dibawah ini.



Gambar 2.4 Diagram UML

Berikut ini penjelasan singkat dari pembagian kategori tersebut.

1. *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.

2. *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
3. *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.

2.11.2.*Class* Diagram

Rosa A.S dan M. Shalahuddin (2013), diagram kelas atau *class diagram* merupakan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

1. Atribut merupakan variable-variabel yang dimiliki oleh suatu kelas.
2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Pada kelas dibuat agar pembuat program atau *programmer* membuat kelas-kelas sesuai rancangan didalam kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron. Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem sehingga pembuat perangkat lunak atau *programmer* dapat membuat kelas-kelas didalam program perangkat lunak sesuai dengan perancangan diagram kelas. Sususan kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis berikut :

1. Kelas main
   Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
2. Kelas yang menangani tampilan sistem (*view*)
   Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.
3. Kelas yang diambil dari pendefinisian *use case* (*controller*)
   Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*, kelas ini biasanya disebut dengan kelas proses yang menangani proses bisnis pada perangkat lunak.

4. Kelas yang diambil dari pendefinisian data (*model*)

   Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data. Semua tabel yang dibuat di basis data dapat dijadikan kelas, namun untuk tabel dari hasil relasi atau atribut multivalue ERD dapat dijadikan kelas tersendiri dapat juga tidak asalkan pengaksesannya dapat dipertanggungjawabkan atau tetap di dalam perancangan kelas.

Rosa A.S dan M. Shalahuddin (2013), simbol-simbol yang ada pada diagram kelas (*class diagram*) dapat dilihat pada Tabel 2.1 berikut.

Tabel 2.1 Simbol *Class Diagram*

Simbol	**Deskripsi**
Kelas **nama_kelas** +atribut -operasi()	Kelas pada struktur system
Antarmuka/*interface* nama_interface	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
Asosiasi/*assocication*	Relasi antar kelas dengan makna umum, aosiasi biasanya juga disertai dengan *multiplicity*
Asosiasi berarah/*directed assocication*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
Generalisasi	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus)
Kebergantungan/*dependency*	Relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi/*aggregation*	Relasi antar kelas dengan makna semua-bagian (*whole-part*)

2.11.3. *Use Case* Diagram

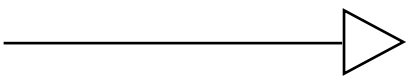
Rosa A.S dan M. Shalahuddin (2013), *use case* atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi satu atau lebih aktor dengan sistem informasi yang akan dibuat. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case.*

a. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.

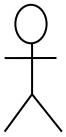
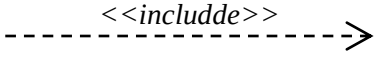
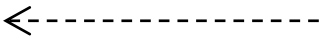
b. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Rosa A.S dan M. Shalahuddin (2013), simbol-simbol yang ada pada diagram *use case* dapat dilihat pada Tabel 2.2 berikut.

Tabel 2.2 Simbol *Use Case*

Simbol	Deskripsi
Use case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *use case.*
Asosiasi/*association* 	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
Generalisasi/*generalization* 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.

Tabel 2.2 (Lanjutan)

Aktor/*actor*  nama actor	Orang, proses, atau sistem lain yang berinterkasi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda diawal frase nama aktor.
Menggunakan *include* <<*includde*>> 	Relasi *use case* tambahan ke sebuah *use case* dimana use case yang ditambahkan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini ada dua sudut pandang yang cukup besar mengenai include di *use case* : 1) include berarti *use case* yang ditambahkan akan selalu dipanggil saat *use case* tambahan dijalankan, 2) include berarti *use case* yang tambahan akan selalu melakukan pengecekan apakah *use case* yang ditambahkan telah dijalankan sebelum *use case* tambahan dijalankan.
Ekstensi/*extend* <<*extend*>> 	Relasi *use case* tambahan ke *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu; mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan.

Use case nantinya akan menjadi kelas proses pada diagram kelas sehingga perlu dipertimbangkan penamaan yang dilakukan apakah sudah layak menjadi kelas atau belum sesuai dengan aturan pendefinisian kelas yang baik.

2.11.4. *Activity* Diagram

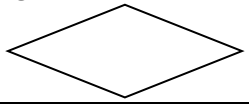
Rosa A.S dan M. Shalahuddin (2013), diagram aktivitas atau *activity diagram* menggambarkan *workflow* (alir kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Diagram aktivitas juga banyak digunakan untuk mendefiniskan hal-hal berikut :

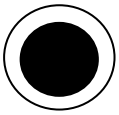
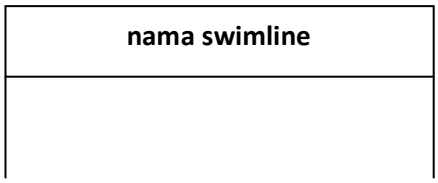
a. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
b. Urutan atau pengelompokan tampilan dari sistem/*uses interface* diaman setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
c. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
d. Rancangan menu yang ditampilkan pada perangkat lunak.

Rosa A.S dan M. Shalahuddin (2013), simbol-simbol yang ada pada diagram aktivitas dapat dilihat pada Tabel 2.3 berikut ini.

Tabel 2.3 Simbol *Activity Diagram*

Simbol	Deskripsi
Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah sistem awal.
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
Simbol	**Deskripsi**
Percabangan/*decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.

Tabel 2.3 (Lanjutan)

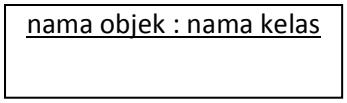
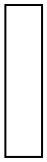
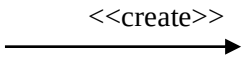
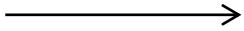
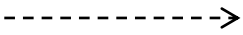
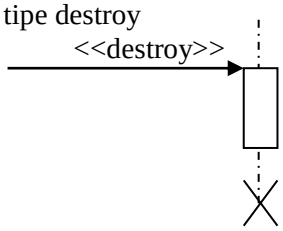
Penggabungan/*join* 	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
Swimlame 	Memisahkan organisasi bisnis yang bertanggungjawab terhadap aktivitas yang terjadi.

2.11.5. *Sequence* Diagram

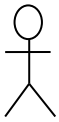
Rosa A.S dan M. Shalahuddin (2013), diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat scenario yang ada pada *use case.* Banyaknya diagram *sequen* yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak.

Rosa A.S dan M. Shalahuddin (2013), simbol-simbol yang ada pada diagram sekuen dapat dilihat pada Tabel 2.4 berikut.

Tabel 2.4 Simbol *Sequence Diagram*

Simbol	Deskripsi
Garis hidup/*lifeline*	Menyatakan kehidupan suatu objek.
Objek 	Menyatakan objek yang berinteraksi pesan.
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya.
Pesan tipe create 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat.
Pesan tipe call 	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri.
Pesan tipe send 	Merupakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim.
Pesan tipe return 	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.
Pesan tipe destroy 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhirim sebaliknya jika ada create maka ada destroy.

Tabel 2.4 (Lanjutan)

Aktor  atau nama_aktor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.

Penomoran pesan berdasarkan urutan interaksi pesan. Penggambaran letak pesan harus berurutan, pesan yang lebih atas dari lainnya adalah pesan yang berjalan terlebih dahulu. Semua metode di dalam kelas harus ada di dalam diagram kolaborasi atau sekuen, jika tidak ada berarti perancangan metode di dalam kelas itu kurang baik. Hal ini dikarenakan ada metode yang tidak dapat dipertanggungjawabkan kegunaannya.