BAB II

TINJAUAN PUSTAKA

II.1. Multimedia

Multimedia adalah pemanfaatan komputer untuk membuat dan menggabungkan teks, grafik, audio, gambar bergerak (video dan animasi) dengan menggabungkan *link* dan *tool* yang memungkinkan pemakai melakukan navigasi, berinteraksi, berkreasi dan berkomunikasi. Dalam definisi ini terkandung empat komponen penting multimedia. Pertama, harus ada komputer yang mengkoordinasikan apa yang di lihat dan di dengar, yang berinteraksi dengan kita. Kedua, harus ada *link* yang menghubungkan kita dengan informasi. Ketiga, harus ada alat navigasi yang memandu kita, menjelajah jaringan informasi yang saling terhubung. Keempat, multimedia menyediakan tempat kepada kita untuk mengumpulkan, memproses, dan mengkomunikasikan informasi dan ide kita sendiri. Jika salah satu komponen tidak ada, maka bukan multimedia adalam arti yang luas namanya. Misalnya, jika tidak ada komputer untuk berinteraksi, maka itu namanya media campuran, bukan multimedia. Jika tidak ada link yang menghadirkan sebuah struktur dan dimensi, maka memungkinkan kita memilih jalannya suatu tindakan maka itu namanya film, bukan multimedia (M. Suyanto ; 2007 :21).

II.2. Pengertian Aplikasi

Program aplikasi adalah program siap pakai atau program yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain. Aplikasi juga diartikan sebagai penggunaan atau penerapan suatu konsep yang menjadi pokok pembahasan atau sebagai program komputer yang dibuat untuk menolong manusia dalam melaksanakan tugas tertentu. Aplikasi *software* yang dirancang untuk penggunaan praktisi khusus, klasifikasi luas ini dapat dibagi menjadi 2 (dua) yaitu:

a. Aplikasi *software* spesialis, program dengan dokumentasi tergabung yang dirancang untuk menjalankan tugas tertentu.
b. Aplikasi paket, suatu program dengan dokumentasi tergabung yang dirancang untuk jenis masalah tertentu (Rahmatillah ; 2011 : 3).

II.3. Pengertian Game

Game adalah salah satu media yang berkembang sangat pesat 2 dasawarsa belakangan ini, mengalahkan laju perkembangan media film. Di awal 1990-an game masih bisa dianggap sebagai komoditas anak-anak, di era 2000 ke atas, game sudah menjangkau berbagai bidang seperti hiburan untuk semua kalangan, bisnis, simulasi, edukasi dan juga pembelajaran virtual. Modal atau perputaran uang dalam industri game sangat besar dan pertumbuhannya sangat pesat.

Tidak hanya di negara Barat sebagai pusat industri, kini pertumbuhan game, baik dari segi produsen maupun konsumen, juga mulai merambah pasar Asia. Bahkan asia memegang prestasi pertumbuhan pengguna game di tahun 2008

dan diramalkan akan tetap memegang posisi tersebut di tahun 2008. Tidak dapat dipungkiri lagi bahwa game sudah menjadi salah satu budaya dan keseharian kita. Perlahan budaya game menjadi gaya hidup yang banyak di kenal dan diakrabi oleh generasi mudah terlebih di daerah perkotaan. Kini, sudah banyak anak umur 3 tahun yang mengenal berbagai jenis perangkat game dari model sederhana seperti *gamewatch* sampai yang canggi seperti konsol *playstation* atau jenis komputer.

Kalau pada tahun 1990 an masih terbatas penyebarannya, kini komputer sudah marak. Bahkan di sekolah dasar pun sudah banyak tersedia laboratorium komputer. Memang harus diakui bahwa spesifikasi komputer di sana kurang memadai, tapi kita melihat bahwa penetrasi teknologi sudah mencapai usia dini dan diakui oleh pemerintah dan para pelaku pendidikan sebagai salah satu penunjang keahlian siswa.

Walaupun mungkin di mata anda *game* umumnya hanya dimainkan oleh kalangan muda, sebenarnya banyak juga kalangan dewasa yang memainkannya di kala senggang. Kalau ditilik dari umurnya, mereka adalah generasi pertama pengguna game di era 1980 an yang mengenal game dalam bentuk teknologi sederhana. Jadi, saat ini mereka masih memainkan game, namun bukan dengan teknologi terbaru sehingga mereka bisa dikatakan embrio generasi gamer. Sifat mereka tentu berbeda dengan generasi 1990 an atau 2000 an sudah lebih tinggi, baik dari teknologi maupun keterikatannya dengan game (Samuel Henry ; 2010 : 8).

II.4. Flash

Flas merupakan sebuah program yang ditujukan kepada para desainer atau programmer yang bertujuan merancang animasi untuk pembuatan sebuah hakaman web, pembuatan game onteraktif, presentasi untuk tujuan bisnis, proses pembelajaran, pembuatan film kartun, dan dapat digunakan untuk membangun sebuah aplikasi yang bernilai tinggi serta tujuan lain yang lebih spesifik lagi.

Teknologi flash menjadi solusi bagi penyebar informasi atau pembangun aplikasi untuk disebarkan ke khalayak ramai sehingga menjadi teknologi yang populer dan berkembang dengan pesat. Flash dapat dilihat dari dua aspek, yaitu :

1. Flash sebagai *software.* Macromedia flash sebagai *software* pembuat atau pembangun aplikasi, sistem informasi dan pembuat animasi.
2. Flash sebagai teknologi. Sekrang ini hampir semua *browser* serta sebagian peralatan eletronik sudah terinstal flash player untuk dapat menjalankan animasi.

Flash adalah program animasi yang berbasis vektor yang dapat menghasilkan file yang berukuran kecil sehingga mudah diakses. Flash dilengkapi dengan tool untuk membuat gambar yang kemudian akan dibuat animasi atau akan dijalankan dengan *script*nya (*script* dalam flash disebut *actionscript*).

Kelebihan dari flash adalah dengan adanya bahasa *scripting* yang disebut *actionscript* yang mempunyai kemampuan yang sudah tidak diragukan lagi untuk mendukung perancangan suatu animas atau aplikasi yang sederhana sampai yang komplek sekalipun (Teguh Wahyono ; 2009 : 2).

II.5. Teknik Frame

Bentuk dasar dari animasi adalah animasi frame per frame. Animasi frame per frame menuntut banyak gambar yang harus dibuat. Efek animasi diciptakan dengan mengganti gambar yang satu dengan gambar yang lain selama beberapa waktu. Contoh animasi frame per frame dapat dilihat pada Help | Samples | Flower. Semua gambar yang bergerak dihasilkan dari gambar yang berbeda-beda tiap framenya. Karena animasi frame per frame harus memiliki gambar yang unik tiap framenya maka animasi frame per frame sangat ideal untuk membuat animasi yang kompleks yang terdiri dari banyak perubahaan seperti ekspresi wajah. Kelemahan dari animasi frame per frame adalah membutuhkan banyak waktu untuk membuat setiap gambar dan menghasilkan file yang besar ukurannya.

Di dalam Flash, sebuah frame yang memiliki gambar yang unik dinamakan keyframe. Animasi frame per frame membutuhkan gambar yang unik setiap framenya, hal ini menyebabkan setiap framenya adalah keyframe. Bentuk dasar dari animasi adalah animasi frame per frame. Animasi frame per frame menuntut banyak gambar yang harus dibuat. Efek animasi diciptakan dengan mengganti gambar yang satu dengan gambar yang lain selama beberapa waktu. Contoh animasi frame per frame dapat dilihat pada Help | Samples | Flower. Semua gambar yang bergerak dihasilkan dari gambar yang berbeda-beda tiap framenya. Karena animasi frame per frame harus memiliki gambar yang unik tiap framenya maka animasi frame per frame sangat ideal untuk membuat animasi yang kompleks yang terdiri dari banyak perubahaan seperti ekspresi wajah. Kelemahan dari animasi frame per frame adalah membutuhkan banyak waktu untuk membuat setiap gambar dan menghasilkan file yang besar ukurannya.

Di dalam Flash, sebuah frame yang memiliki gambar yang unik dinamakan keyframe. Animasi frame per frame membutuhkan gambar yang unik setiap framenya, hal ini menyebabkan setiap framenya adalah keyframe.

Untuk menambah keyframe pada timeline, sorot frame dan kemudian lakukan satu dari beberapa cara berikut ini :

1. Klik kanan dan pilih Insert Keyframe.
2. Pilih Insert | Keyframe dari menu.
3. Tekan F6 pada keyboard.

Di bawah ini akan ditunjukkan proses pembuatan animasi frame per frame:

1. Sorot frame dimana akan memulai animasi frame per frame.
2. Jikalau belum berupa sebuah keyframe, buatlah menjadi keyframe dengan cara memilih menu Insert | Keyframe (F6).
3. Gambarlah atau impor gambar pertama dari urutan gambar pada keyframe ini.
4. Kemudian klik frame selanjutnya dan buatlah menjadi keyframe yang lain. Rubahlah isi dari keyframe ini dengan gambar berikutnya.
5. Lakukan penambahan keyframe dan perubahan isinya sampai memperoleh animasi yang utuh. Pada akhirnya teslah animasi dengan kembali ke frame pertama dan kemudian pilih menu Control | Play (Ali Salim ; 2013 : 1).

II.6. Tweening

Tween animation sangat mengurangi waktu karena tidak perlu membuat animasi secara frame per frame. Sebaliknya hanya membuat frame awal dan

frame akhir saja. Dua alasan utama mengapa *tween animation* sangat baik yaitu karena mengurangi pekerjaan mengambar dan meminimalkan ukuran file karena isi dari setiap frame tidak perlu disimpan.

Ada 2 jenis tween animation yaitu Shape tween dan Motion tween, dimana masingmasing memiliki karakter yang unik.

1. Shape Tweening (Animasi Perubahan Bentuk)

Shape tweening berguna untuk mengubah bentuk. Flash hanya dapat mengubah bentuk, jadi jangan mencoba untuk melakukan *Shape tween* untuk *group*, symbol, atau teks. Hal ini tidak akan berhasil. dapat melakukan *Shape Tween* pada beberapa bentuk di dalam sebuah layer, tetapi lebih baik untuk menempatkannya pada layer yang berbeda.

Hal ini akan memudahkan jika kita ingin melakukan perubahan. *Shape tweening* juga memperbolehkan untuk mengubah warna. Langkah-langkah membuat *shape tween* adalah :

a. Sorot frame di mana ingin membuat animasi. Jika belum berupa sebuah keyframe maka rubahlah menjadi keyframe.
b. Buatlah gambar pada *stage*. Ingatlah *shape tween* hanya bekerja pada bentuk – bukan pada group, simbol atau teks yang dapat diedit. Untuk 3 elemen tersebut maka perlu mengubahnya menjadi bentuk dasar dengan melakukan perintah Modify | Break Apart.
c. Buatlah sebuah keyframe kedua, dan buatlah gambar akhir pada *stage*.

d. Bukalah kotak dialog Frame Properties dengan mengklik g pada frame mana pun yang berada di antara dua keyframe. dapat juga menyorot sebuah frame di antaranya dan kemudian memilih menu Modify | Frame.
e. Pilih tab Tweening dan pilih Shape pada *field* Tweening. akan melihat beberapa pilihan untuk merubah *shape*.
f. Sorot salah satu Blend Type. Pilihan Distributive akan menciptakan animasi transasi bentuk yang halus, sedangkan pilihan Angular akan menciptakan transisi bentuk dengan sudut dan garis-garis lurus. Jika gambar akhir berisi bentuk yang mempunyai sudut dan garis, pilih Angular, selain itu pilih Distributive.
g. Jika perlu rubahlah Easing slider. Easing menentukan bagaimana animasi dari awal sampai akhir. Ini berguna jika ingin menciptakan efek mempercepat atau memperlambat. Jika ingin anmiasi mulai dengan lambat dan kemudian cepat, geser slider ke arah in. Untuk animasi yang permulaannya cepat, dan kemudian melambat, geser slider ke arah out. Jika ingin kecepatan animasi konstan maka biarkan slider berada di tengah. dapat juga mengetik besar dari Easing yaitu dari –100 sampai 100.
h. Tekan tombol OK. Teslah animasi dengan memilih menu Control | Play (Enter).

2. Shape hints

Shape hints memberikan kontrol untuk *shape tween* yang kompleks. bisa mengatur setiap titik awal dari animasi perubahan bentuk lalu titik akhir dari

animasi perubahan bentuk tersebut. Dengan memberi beberapa *shape hints* maka kita bisa mengontrol animasi perubahan bentuk sesuai dengan yang kita inginkan karena dalam animasi perubahan bentuk yang kompleks seringkali apa yang kita inginkan tidak bisa dilakukan oleh program Flash secara otomatis kecuali melalui *shape hints*.

dapat dengan mudah menggunakan *shape hints* pada *shape tween* dengan cara sebagai berikut :

a. Buatlah *shape tween* dengan menggunakan langkah-langkah Membuat Shape Tween di atas.
b. Sorot frame awal dari *shape tween*. Pilih Modify | Transform | Add Shape Hint, atau tekan Ctrl + H untuk menambah sebuah *shape hint*. *Shape hint* muncul dengan lingkaran merah dengan sebuah huruf di dalamnya (huruf itu mulai dari a sampai z).
c. Pindahkan *shape hint* ke tempat yang inginkan.
d. Sekarang sorot frame terakhir. akan melihat lingkaran hijau kecil dengan huruf yang sama seperti pada *shape hint* awal. Pindahkan *shape hint* ke tempat di mana ingin *shape hint* pertama bergerak menujunya.
e. Jalankan *movie* (Control | Play) untuk melihat bagaimana *shape hint* mempengaruhi *tweening*.
f. Tambahkan *shape hint* hingga puas dengan hasilnya. Ingatlah untuk mencocokkan *shape hint* yang ada pada frame awal dan frame akhir – *a* harus ke *a*, *b* ke *b* dan seterusnya.

Jika setelah menambahkan *shape hint* kemudian tidak menginginkan *shape hint* tersebut, dapat membuang semua *shape hint* dengan memilih menu Modify | Transform | Remove All Hints. dapat juga mengklik kanan pada salah satu shape hint untuk membuka menu pop-up *shape hint*. Menu ini memungkinkan untuk menambahkan sebuah hint (Add Hint), membuang sebuah hint (Remove Hint), atau Remove All Hint (Ali Salim ; 2013 : 2).

II.7. Motion Tweening (Animasi gerak)

Motion tween tidak hanya berguna untuk menggerakkan *groups*, simbol, atau teks yang dapat diedit dari satu tempat ke tempat lain. *Motion tween* menolong untuk merubah ukuran, memutar, merubah warna dan transparansi simbol. *Motion tween* hanya bisa digunakan pada satu objek pada satu layer. Jadi jika ingin mengerakkan banyak objek maka membutuhkan banyak layer. Langkah-langkah membuat sebuah *motion tween* adalah :

1. Sorot frame di mana ingin membuat animasi. Jika belum berupa sebuah keyframe maka rubahlah menjadi keyframe.
2. Gambar atau imporlah gambar yang ingin lakukan animasi. hanya dapat melakukan animasi pada group, simbol, dan teks yang dapat diedit.
    a. Jika menggunakan sebuah gambar, grouplah gambar tersebut atau ubah menjadi sebuah simbol.
    b. Jika mempunyai gambar berupa simbol di *library*, cukup men*drag* gambar dari *library* ke *stage*.

c. Jika menggunakan teks yang dapat diedit, tidak perlu melakukan apa pun karena sudah berupa objek.

3. Sorot frame dimana animasi akan berakhir. Rubah frame ini menjadi sebuah keyframe dengan memilih menu Insert | Keyframe.
4. Pindahkanlah gambar dalam stage ke tempat yang ingin dituju. Ingatlah tidak hanya dapat menggerakkan gambar, juga bisa melakukan putaran, merubah ukuran, dan merubah efek warna.
5. Klik kanan pada sebuah frame di antara kedua keyframe dan pilih Create Motion Tween. Teslah animasi .
6. Jika ingin mengubah *properties motion tween*, klik g pada salah satu frame yang berada di antara kedua keyframe untuk membuka kotak dialog Frame Properties. Pilih tab Tweening. dapat melakukan perubahan *properties*,
   a. Tween scaling – Jika ingin mengubah ukuran.
   b. Rotate – Jika ingin memutar objek. Pilih jenis putaran pada menu *dropdown* dan kemudian angka rotasi pada kotak times. Jika mengetik 0 pada kotak times, atau memilih None pada menu *drop-down*, tidak ada putaran yang dilakukan.
   c. Orient to path direction – Ketika objek mengikuti sebuah alur, pilihan ini menyebabkan objek berada di tengah alur.
   d. Easing. Easin – menentukan bagaimana animasi dari awal sampai akhir. Ini berguna jika ingin menciptakan efek mempercepat atau memperlambat. Jika ingin anmiasi mulai dengan lambat dan kemudian cepat, geser slider ke arah in. Untuk animasi yang permulaannya cepat,

dan kemudian melambat, geser slider ke arah out. Jika ingin kecepatan animasi konstan maka biarkan slider berada di tengah. dapat juga mengetik besar dari Easing yaitu dari –100 sampai 100 (Ali Salim ; 2013 : 3).

II.8. *Flowchart Of Document (FOD)*

Flowchart adalah penggambaran secara grafik dari langkah-langkah dan urut-urutan prosedur dari suatu program. Flowchart menolong analis dan programmer untuk memecahkan masalah kedalah segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam pengoperasian.

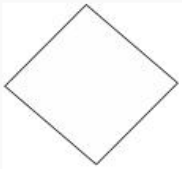
Flowchart biasanya mempermudah penyelesaian suatu masalah khususnya masalah perlu dipelajari dan dievaluasi lebiih lanjut. Bila seorang analis dan programmer akan membuat flowchart, ada beberapa petunjuk yang harus diperhatikan, seperti :

1. Flowchart digambarkan dari halaman atas ke bawah dan dari kiri ke kanan.
2. Aktivitas yang digambarkan harus didefinisikan secara hari-hati dan definisi ini harus dapat dimengerti oleh pembacanya.
3. Kapann aktivitas dimulai dan berakhir harus ditentukan secara jelas.
4. Setiap langkah dari aktivitas harus diuraikan dengan menggunakan deskripsi kata kerja.
5. Setiap langkah dari aktivitas harus berada pada urutan yang benar.

6. Lingkup dan range dari aktifitas yang sedang digambarkan harus ditelusuri dengan hari-hati.
7. Gunakan simbol-simbol flowchart yang str.

Flowchart paperwork menelusuri alur data yang ditulis melalui sistem flowchart paperwork sering disebut juga dengan flowchart dokumen. Kegunaan utamanya adalah untuk menelusuri alur form dan laporan sistem dari satu bagian ke bagian lain baik bagaimana alur form dan laporan diprose, dicatat dan disimpan (Sulindawati ; 2010 : 8).

Tabel III.1. Simbol *Flowchart*

Gambar	Simbol untuk	Keterangan
	Proses / Langkah	Menyatakan kegiatan yang akan ditampilkan dalam diagram alir.
	Titik Keputusan	Proses / Langkah dimana perlu adanya keputusan atau adanya kondisi tertentu. Di titik ini selalu ada dua keluaran untuk melanjutkan aliran kondisi yang berbeda.
	Masukan / Keluaran Data	Digunakan untuk mewakili data masuk, atau data keluar.
	Terminasi	Menunjukkan awal atau akhir sebuah proses.
	Garis alir	Menunjukkan arah aliran proses atau algoritma.
	Kontrol / Inspeksi	Menunjukkan proses / langkah dimana ada inspeksi atau pengontrolan.

Sumber : Sulindawati ; 2010 : 8

II.14. UML (*Unified Modeling Language*)

Menurut Sri Dharwiyanti (2013) *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.

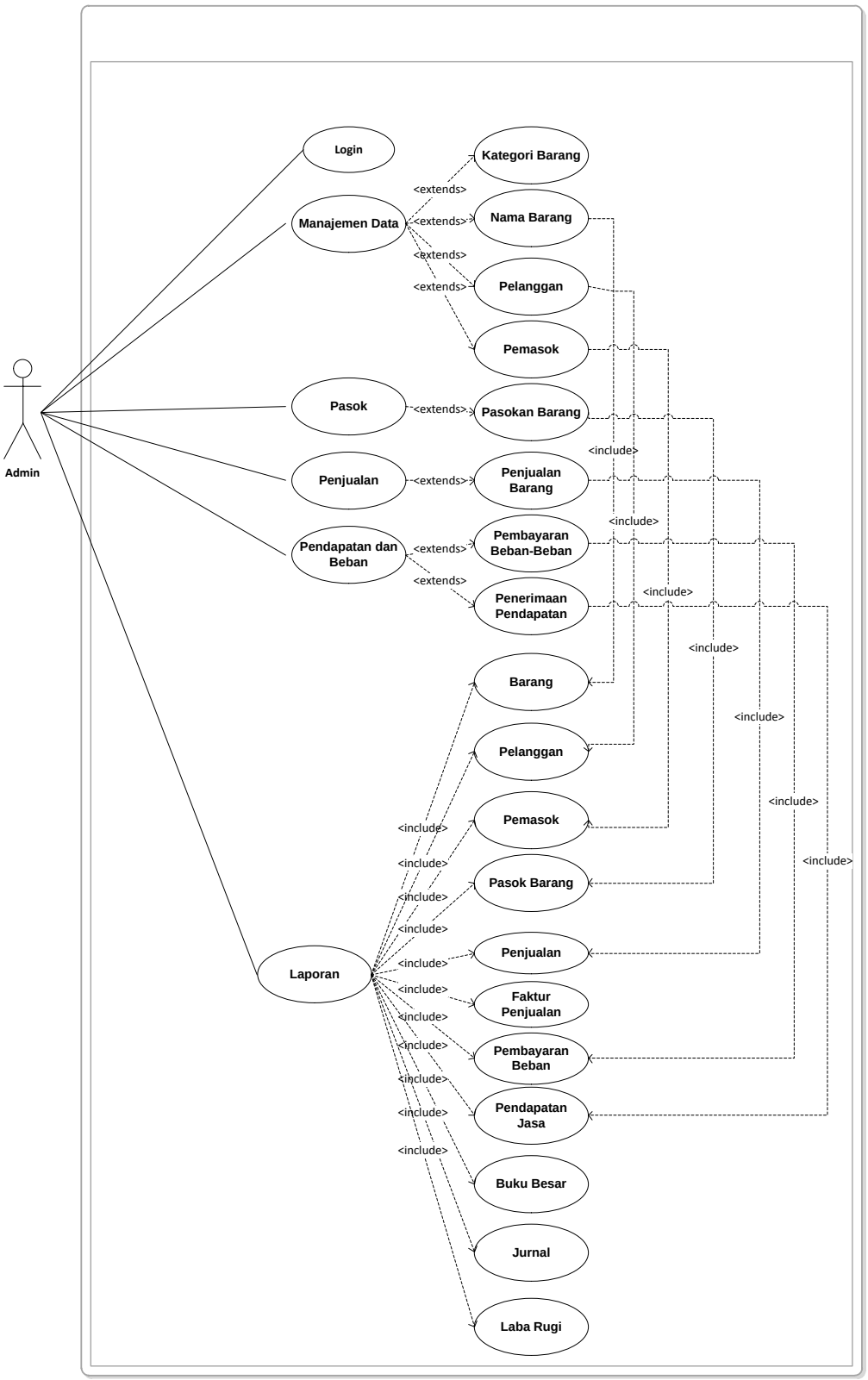
Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep 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.

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 system 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.

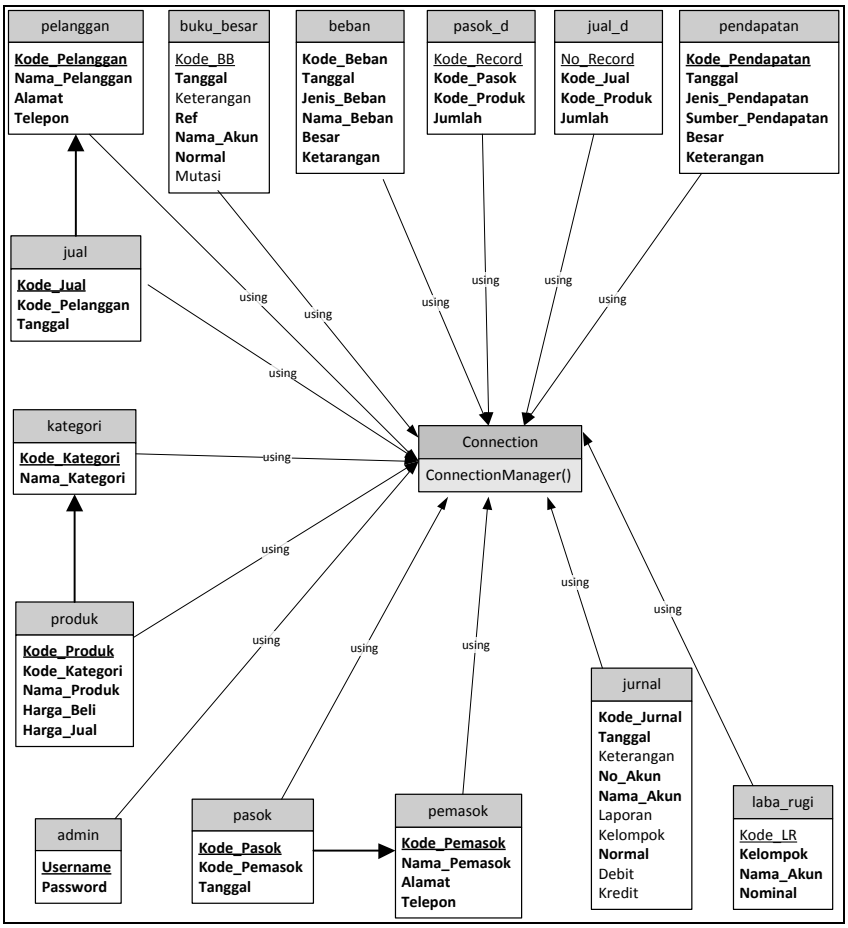


Gambar II.2. *Usecase* Diagram
(Sumber : Sri Dharwiyanti ; 2013 : 5)

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.

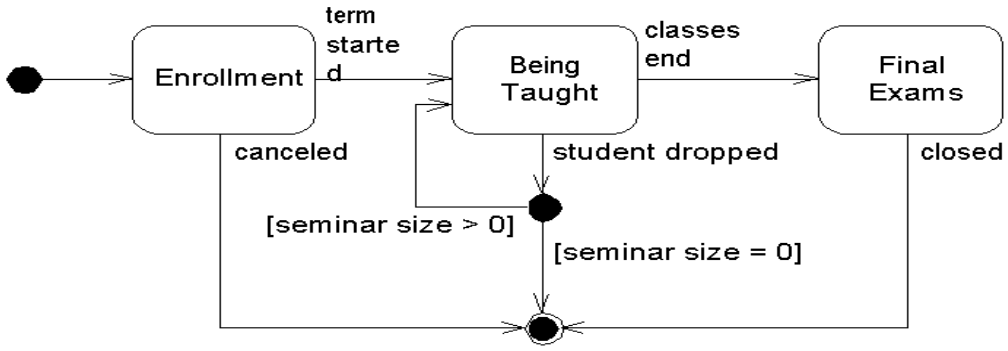


Gambar II.3. *Class* Diagram
(Sumber : Sri Dharwiyanti ; 2013 : 6)

3. *Statechart* Diagram

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).

Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring. Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.



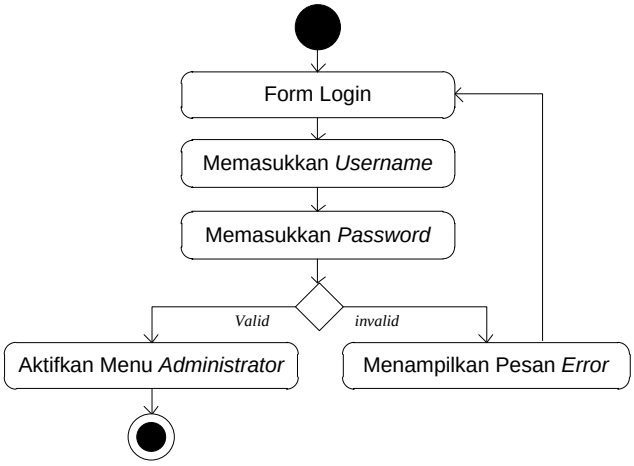
Gambar II.4. *Statechart* Diagram
(Sumber : Sri Dharwiyanti ; 2013 : 7)

4. *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, 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



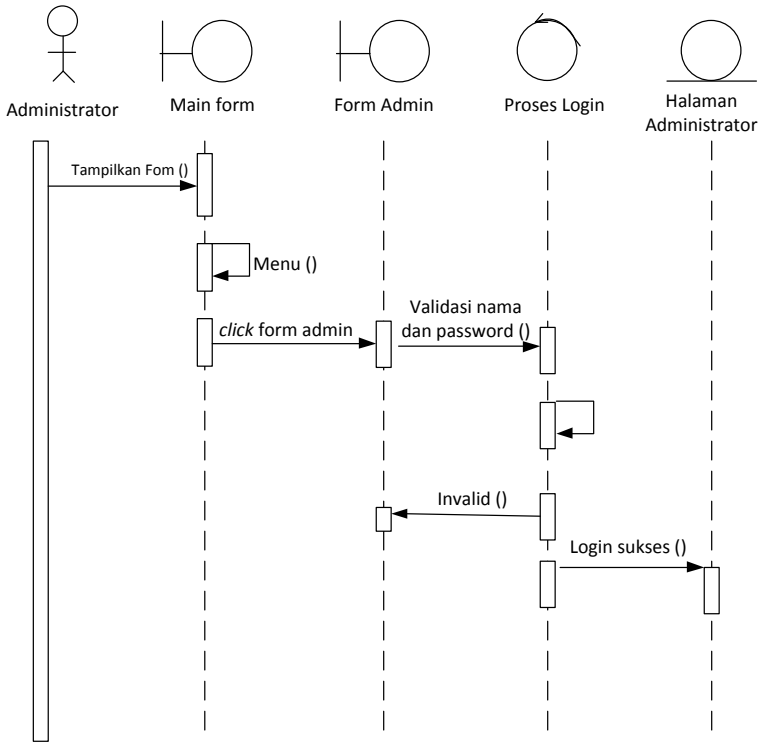
Gambar II.5. *Activity* Diagram
(Sumber : Sri Dharwiyanti ; 2013 : 8)

5. *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.

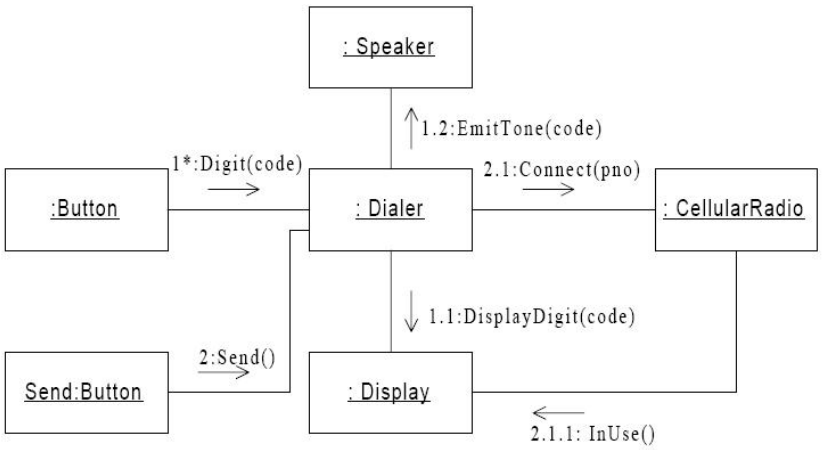


Gambar II.6. *Sequence* Diagram
(Sumber : Sri Dharwiyanti ; 2013 : 9)

6. *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.

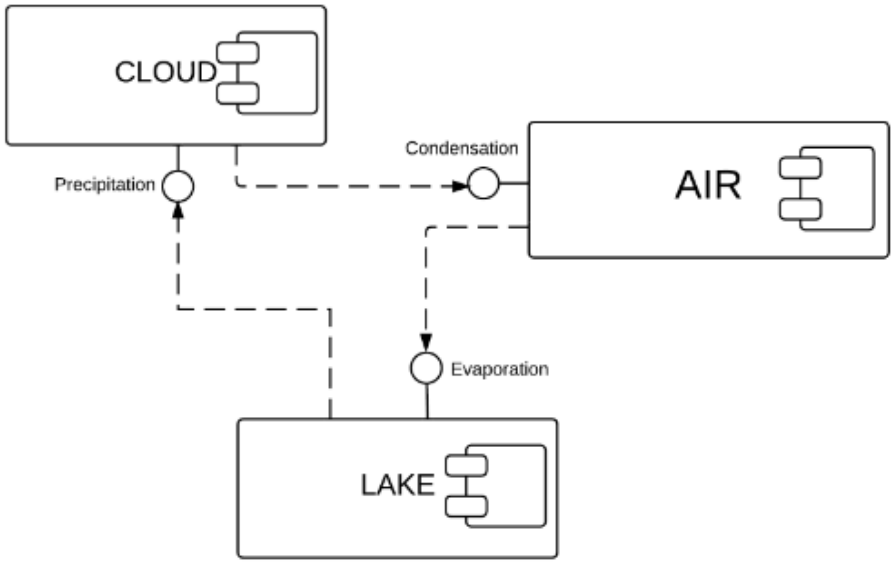


Gambar II.7. *Collaboration* Diagram
(Sumber : Sri Dharwiyanti ; 2013 : 9)

7. *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.

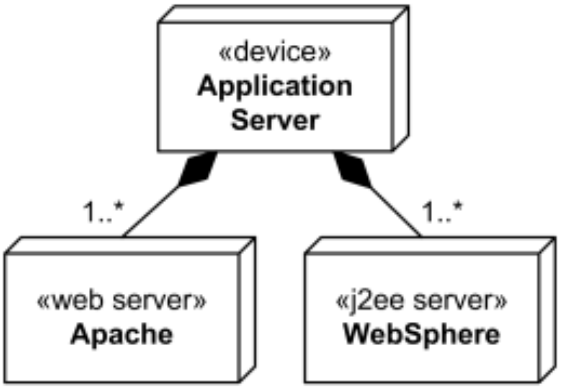
Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.



Gambar II.8. *Component* Diagram
(Sumber : Sri Dharwiyanti ; 2013 : 10)

8. *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.



Gambar II.9. *Deployment* Diagram
(Sumber : Sri Dharwiyanti ; 2013 : 11)