BAB II

LANDASAN TEORI

II.1. Konsep Dasar *Game*

II.1.1. Pengertian *Game*

Game berasal dari bahasa inggris yang memiliki arti dasar permainan. Permainan dalam hal ini merunjuk pada pengertian "kelincahan intelektual" (*Intellectual Playability*). *Game* juga bisa diartikan sebagai arena keputusan dan aksi permainannya. Permainan juga dapat disebut sesuatu yang dimainkan dengan aturan tertentu sehingga ada yang menang dan ada yang kalah, biasanya dalam konteks tidak serius atau dengan tujuan *refreshing* (hiburan).

Menurut Ernest Adams dalam bukunya "*Fundamentals of Game Design, 2.nd Edition*" (2010, 2), *Game* adalah salah satu jenis aktifitas bermain, yang didalamnya dilakukan dalam konteks berpura-pura namun terlihat seperti realitas, yang mana pemainnya memiliki tujuan untuk mendapatkan satu kemenangan serta dilakukan dengan sesuai dengan aturan permainan yang dibuat.

Game berasal dari keinginan untuk bermain (*playing*) dan kemampuan untuk berpura-pura (*pretending*) yang dimiliki manusia. Bermain merupakan aktifitas manusia yang termasuk golongan *nonessential* dan biasanya untuk tujuan hiburan yang berhubungan dengan kehidupan sosial. Berpura-pura adalah kemampuan mental untuk menciptakan sebuah realitas khayalan (*notional reality*) dimana orang yang berpura-pura tersebut dan menciptakan, mengubah, atau meninggalkan realitas tersebut sesuai keinginannya.

II.1.2. Elemen dalam *Game*

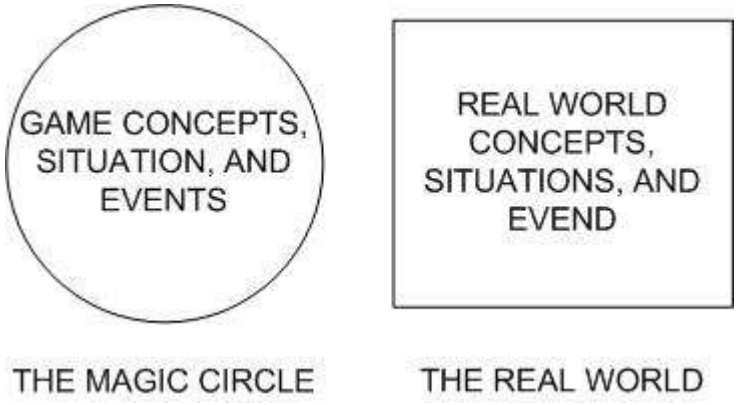
Elemen-elemen penting dalam sebuah *game* adalah bermain (*play*), berpura-pura (*pretend*), tujuan (*goal*), dan peraturan (*rules*) (Ernest Adams, 2010:6-9). Adapun definisi dari elemen-elemn tersebut adalah sebagai berikut:

1. Bermain (*Play*)

Bermain adalah salah satu bentuk hiburan yang membutuhkan keikutsertaan (komunikasi dua arah), dimana buku, *film*, dan teater adalah bentuk hiburan yang berupa pertunjukkan (komunikasi dua arah). Ketika membaca buku, penulis yang menghibur pembaca, tetapi ketika bermain, pemainlah yung menghibur dirinya sendiri. Sebuah buku tidak dapat berubah meskipun sudah sering dibaca, namun ketika bermain, pemain dapat membuat keputusan yang mempengaruhi kejadian dalam permainan tersebut.

2. Berpura-pura (*Pretending*)

Berpura-pura adalah kegiatan untuk menciptakan sebuah realitas khayalan di dalam pikiran. Nama lain dari realitas yang diciptakan dari berpura-pura adalah *magic circle*. Istilah ini pertama kali digunakan oleh sejarahwan Belanda bernama Johan Huizinga dalam bukunya berjudul ”*Homo Ludens*” (Huizinga, 1971) yang berhubungan dengan dunia imajinasi dalam sebuah drama atau novel fiksi. Dalam *game*, *magic circle* mengacu pada batas yang memisahkan ide dan aktifitas yang berarti yang berada dalam *game* dari ide dan aktifitas yang berarti dalam dunia nyata. Dengan kata lain batas antara dunia nyata dan khayalan (*make believe*).



Gambar II.1. Konsep *The Magic Circle* dalam sebuah *game*
(Sumber: Ernest Adams; 2010)

3. Tujuan (*Goal*)

Sebuah *game* harus memiliki satu atau lebih tujuan. Tujuan dari *game* tersebut oleh peraturan dan bentuk *game* itu sendiri karena pembuat *game* dapat menentukannya sesuai dengan keinginanya. Dalam mencapai tujuan dalam *game,* harus ada tantangan. Meskipun kesulitan dari tantangan itu tergantung dari persepsi masing-masing pemain.

4. Peraturan (*Rules*)

Peraturan adalan definisi dan instruksi dimana pemain harus mengiktuinya selama *game* berlangsung. Peraturan memiliki beberapa fungsi yaitu membuat objek *game* dan arti dari setiap aktifitas dan kejadian yang berbeda yang berlangsung di dalam *circle magic*. Peraturan juga membuat pemain mengetahui aktifitas apa yang diperbolehkan dan juga mengevaluasi setiap tindakan yang akan membuat pemain mencapai tujuan dari *game* tersebut.

II.1.3. Jenis *Game*

1. Berdasarkan Jenis (*genre*) Permainannya

Jenis (*Genre*) *game* adalah format atau gaya dari sebuah *game*. Menurut Adams, Ernest (2010, 70-71), *game* memiliki jenis (*genre*) yang luas antara lain:

a. *Actions Games*

b. *Strategy Games*

c. *Role-playing Games*

d. *Real-world Simulations*

e. *Construction and management games*

f. *Adventure Games*

g. *Puzzle Games*

2. Berdasarkan Jenis *Platform* yang Digunakan

Game juga dapat dibedakan berdasarkan *platform* (Mesin *Game*) yang digunakan, menurut Adam, Ernest (2010 , 77-82), adapun jenis *platform* yang digunakan antara lain:

a. *Home Game Consoles*

b. *Personal Computers*

c. *Handheld Game Machines*

d. *Mobile Phones* dan *Wireless Devices*

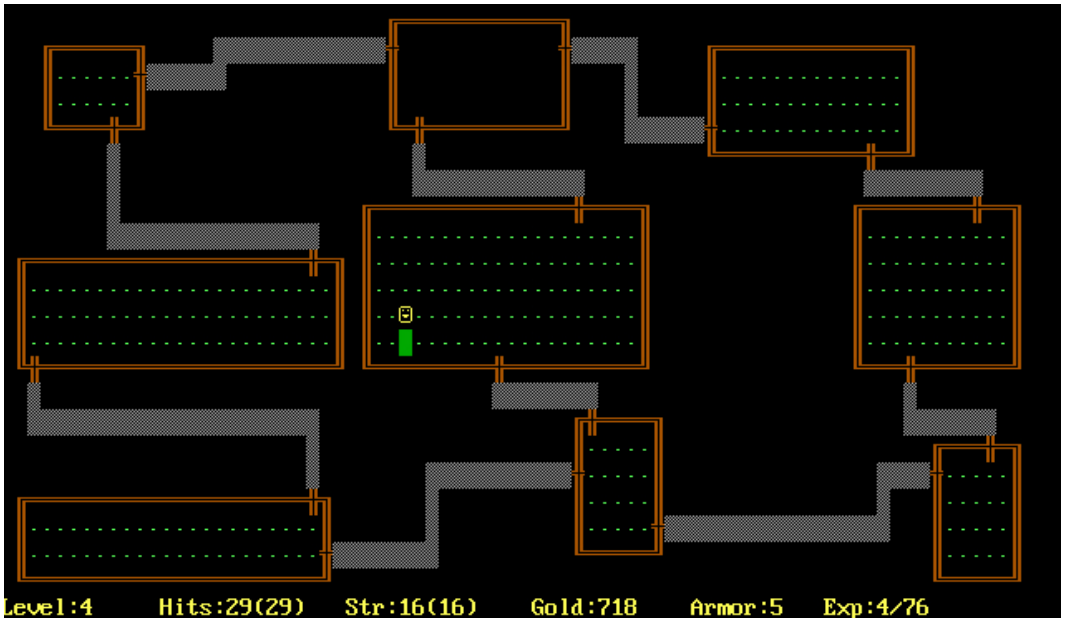
e. Perangkat lainnya seperti *arcade*, mesin judi, dan alat simulasi pesawat.

II.1.4. Definisi *Role-playing Game* (RPG)

Role-playing game (RPG) adalah *game* dimana pemain mengendalikan satu atau lebih karakter, biasanya didesain oleh pemain, dan mengarahkan mereka untuk menyelesaikan beragam misi (*quest*) yang telah diatur oleh komputer. Kemenangan didapat memalui penyelesaian *quest*. Peningkatan kekuatan dan kemampuan karakter merupakan elemen kunci dari *genre* ini. Tantangan berupa pertarungan taktik (*tactical combat*), *logistic*, pertumbuhan ekonomi (*economy growt*), penjelajahan (*explorations*), dan pemecahan teka-teki (*puzzle solving*). Tantangan yang melibatkan koordinasi fisik sangat jarang ada terkecuali dalam RPG-*action hybrid* (Ernest Adams, 2010:445).

II.1.5. Definisi *Roguelike*

Roguelike merupakan *sub-genre* dari RPG (*Role Playing Game)*, ditandai dengan pembentukan level secara prosedural, permainan berbasis *turn-based*, grafis berbasis *tiled*, kematian permanen (*Permanent death/Permadeath*), dan didasarkan pada *setting* cerita fantasi. *Roguelike* juga dapat digambarkan sebagai permainan *turn-based* dengan fokus kepada *gameplay* (teknik permainan) yang rumit dan *replayability* (kemampuan sebuah permainan untuk tidak bosan dimainkan berkali-kali), serta mempresentasikan dunia virtual permainan tersebut menggunakan tampilan berbasis ASCII sebagai ganti dari grafis 3D.



Gambar II.2. Contoh tampilan grafis *game roguelike* "Rogue"
(Sumber: Cliff Jonathan; 2014)

Definisi *roguelike* dibuat di *International Roguelike Development Conference* 2008 dimana definisi itu disebut *Berlin Interpretation*. Dalam *Berlin Interpretation*, terdapat beberapa faktor yang terdapat di dalam sebuah *game roguelike*. Adapun faktor-faktor tersebut antara lain:

1. Faktor Utama
    a. *Random level/enviroment generation* (pembentukan lingkungan/level secara acak), artinya setiap kali pemain memulai permainan baru maka bentuk peta dari lingkungan/level tidak akan sama seperti pada permainan sebelumnya.
    b. *Permadeath* (*permanent death*), apabila karakter mati di tengah permainannya (status HP pemain bernilai 0), maka karakter tersebut akan mati untuk selamanya dan jika pemain memainkan karakter baru maka

semua status dan level akan ter-*reset* beserta *item* yang telah didapat oleh karakter yang sebelumnya juga akan hilang.

c. *Turn-based* (bergiliran), setiap kali pemain melakukan aksi(berjalan satu langkah, menyerang, menggunakan *item*, dan lain-lain) dianggap sebagai satu kali aksi dan musuh akan melakukan aksi setelahnya, lalu kemudian pemain dapat melakukan aksinya kembali.
d. *Grid-base*, lingkungan/level terbentuk dari susunan kotak kecil dimana setiap, *item*, musuh, dan karakter pemain menempati satu kotak tersebut.
e. *Non-modal*, setiap tindakan yang dilakukan pemain berada dalam mode yang sama, artinya pemain dapat melakukan aksi apa saja selama permainan berlangsung.
f. *Complexity* (kerumitan), *game* harus memiliki kerumitan yang memberikan solusi untuk mencapai tujuan. Hal ini diperoleh dengan cara menyediakan *item* dan musuh yang berbeda-beda.
g. *Resource management*, sumber daya yang ada di dalam *game* (seperti makanan, *hp potion*, dan lain-lain) dibuat terbatas agar pemain dapat mengatur persedian yang ada agar karakter tidak mati.
h. *Hack n' slash*, membunuh musuh sebanyak mungkin merupakan bagian penting dalam *roguelike*. Permainan ini merupakan *player vs world game*, dimana dalam *game* ini tidak ada hubungan *enemy-enemy* (seperti diplomasi, atau permusuhan sesama musuh) seperti halnya dalam *game* strategi.

i. *Exploration and discovery* (penjelajahan dan penemuan), *game* ini membutuhkan penjelajahan secara hati-hati dan penemuan *item* untuk mempertahankan nyawa karakter. Dikarenakan pembentukan lingkungan yang acak dan *permadeath*, maka hal ini akan memberikan tantangan bagi pemain.

2. Faktor Pendukung

a. *Single player character*, pemain hanya memainkan satu karakter saja dan kematian dari karakter tersbut merupakan akhri dair permainan.

b. *Enemy's similiarity to player*, musuh juga memiliki kesamaan dengan pemain, seperti memiliki perlengkapan dan *inventory* serta menggunakan *item* dan *magic*.

c. *Tactical challenge*, pemain diharuskan mempelajari taktik sebelum melakukan aksinya untuk mencapai tujuannya.

d. ASCII *display*, *game* ini menggunakan tampilan ASCII untuk mempresentasikan level/lingkungan dari *game* ini.

e. *Dungeon*, lingkungan/level dari *game roguelike* berisi *dungeon* (level yang terdiri dari ruangan dan koridor).

f. *Number*, nomor yang digunakan untuk menjelaskan karakter (seperti *hit point*/darah, *status*, dan lain-lain) ditunjukkan di dalam *game*.

II.2. Pengertian *Game Engine*

Game Engine adalah sistem perangkat lunak yang dirancang untuk menciptakan dan pengembangan *video game*. Ada banyak *game engine* yang

dirancang untuk bekerja pada konsol permainan video dan sistem operasi *desktop* seperti Microsoft Windows, Linux, dan Mac OS X. fungsionalitas inti yang biasanya disediakan oleh *game engine* mencakup mesin render untuk merender grafis 2D atau 3D, mesin fisika atau tabrakan, suara, script, animasi, kecerdasan buatan, jaringan, streaming, manajemen memori, *threading*, dukungan lokalisasi, dan adegan grafik.

II.2.1. *Unity Engine*

Unity Game Engine merupakan *software* yang digunakan untuk membuat video *game* 3D atau konten yang interaktif lainnya seperti, visual arsitektur dan real-time 3D animasi. *Unity Game Engine* tidak hanya merupakan sebuah *game engine*, tapi juga merupakan sebuah *editor*. *Unity Game Engine* mirip dengan *game engine* lainnya seperti, *Director*, *Blender game engine*, *Virtools*, *Torque Game Builder* atau *Gamestudio*.

Kelebihan dari *Unity Game Engine* yaitu *multi platform*, *Unity Game Engine* dapat dioperasikan pada *platform Windows* dan *Mac* OS dan dapat menghasilkan *game* untuk *Windows*, *Mac*, *Linux*, *Wii*, *iPad*, *iPhone*, dan *platform Android*. *Unity Game Engine* juga dapat menghasilkan *game browser*, untuk menjalankan pada *web broser* kita memerlukan sebuah *plugin* yaitu *Unity web player plugin*.

II.2.2. Komponen-komponen *Unity Engine*

1. *Scene*

Sebuah *game* yang dibuat menggunakan *Unity Engine* terdiri dari satu atau lebih *scene*. Dalam *game engine* lain, *scene* bisa juga disebut level (Philip Chu, 2009, 5).

2. *Game Object*

Sebuah *scene* terdiri dair *game object* yang juga disebut entitas (entities) dala *game engine* lain. Sebuah *game object* memiliki nama, posisi, dan orientasi di dalam *scene* dan *attributes* atau *behaviour* tergantung tipe dari objek yang direprsentasikannya.

Game object juga dapat memiliki hubungan *parent-child* dengan objek lainnya. Posisi dan orientasi dari masing-masing *game object* berhubungan dengan *parent*-nya. Hal ini sering disebut sebagai *scene graph* dalam dunia grafis 3D (Philip Chu, 2009, 6).

3. *Classes*

Unity Engine berorientasi objek, artinya setiap objek memiliki sebuah *class*, dan masing-masing *class* dapat berasal dari "*base*" *class* atau "*parent*" *class*. Sebagai sistem berorientasi objek, *Unity* memiliki sebiah hierarki *class*. *Game object* berasal dari *object class*, begitu juga dengan *class* lainnya.

Gambar II.3. Hierarki *class* dari sebuah objek
(Sumber: Philip Chu; 2009:6)

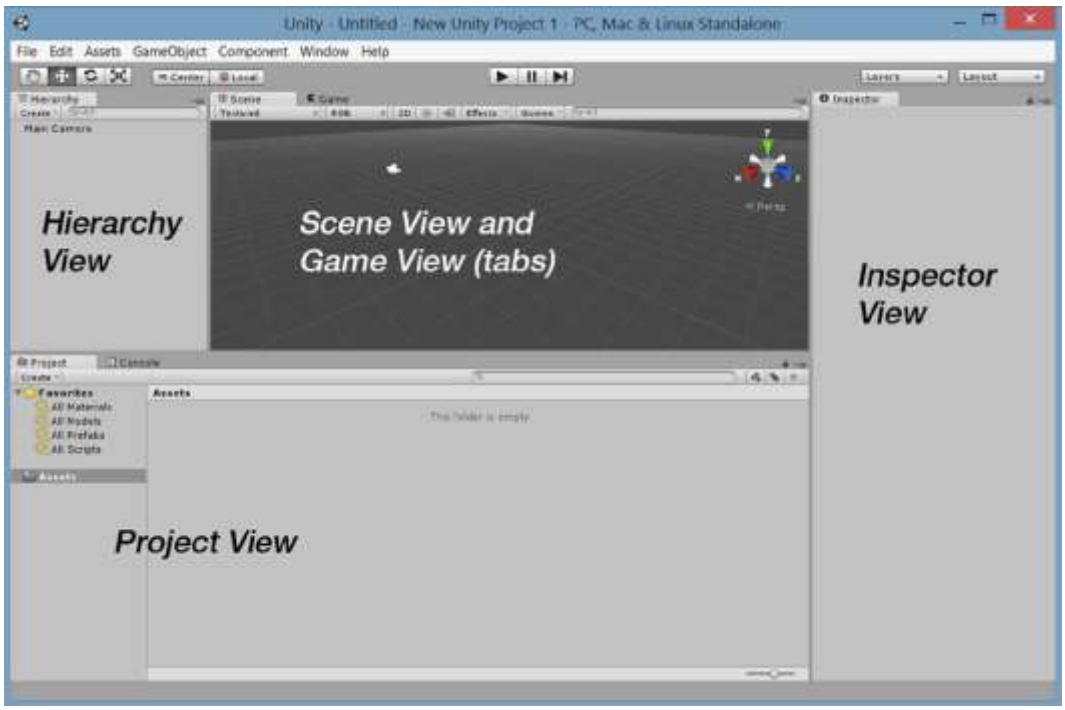
4. *Components*

Dalam *Unity Engine*, setiap node dari *scene graph* merupakan satu *class*, *game object*, dan untuk menentukan *behaviour*-nya, *components* harus ditambahkan sesuai dengan *behaviour* yang diinginkan.

Gambar II.4. Hierarki *class* dari sebuah *component* yang berisi *behaviour*
(Sumber: Philip Chu; 2009: 8)

II.2.3. *Editor Unity Engine*

Game dibuat sebagai *project* dalam *Unity Engine*. *Editor* menampilkan satu *scene Unity* dalam satu waktu. *Editor Unity* terdiri dari 4 tampilan (*view*) utama, yaitu *Scene View*, *Game View*, *Hierarchy View*, dan *Inspector View* (biasanya disebut sebagai *Inspector*) dan beberapa fitur penting seperti *Playback Button*, *Layer Button*, dan lain-lain.



Gambar II.5. Tampilan GUI dari *Editor Unity*
(Sumber: Sue Blackman; 2013:7)

II.2.5. *Scripting*

Scripting (penulisan kode program) dalam *Unity Engine* menggunakan *Mono*, sebuah versi *open-source* dari .NET. *Mono* mendukung banyak bahasa pemrograman, tetapi *Unity* hanya mendukung bahasa C#, *Boo*, dan *Javascript*.

II.2.6. Lingkungan Bahasa Pemrograman *Microsoft Visual* C#

Microsoft Visual C#, yang selanjutnya disebut sebagai C# merupakan bagian dari bahasa keluarga *Microsoft* yang berjalan pada *framework* .NET. C# bebas dari masalah kompabilitas dan dilengkapi dengan berbagai fitur baru, menarik, dan tentu saja menjanjikan (Adi Nugroho, 2010, 1).

C# merupakan bahasa pemrograman yang berorientasi objek dan memiliki banyak kesamaan dengan bahasa C++, *Java*, dan *Visual Basic* (VB). C# merupakan kombinasi antara efisiensi pemrograman C++, kesederhanaan *Java*, dan penyederhanaan dari bahasa VB.

II.3. Pengertian *Android*

Android pertama kali dipublikasikan pada tahun 2005 pada saat *Google* mengakui sisi *Android* Inc. Banyak spekulasi yang menyatakan jika *Google* tertarik untuk memasuki dunia perangkat *mobile*. Pada tahun 2008 *Android* merilis versi 1.0 dan menimbulkan banyak spekulasi, salah satunya adalah bahwa *android* akan menjadi pendatang utama dalam dunia pasar *mobile*. Sejak saat itu *Android* bersaing dengan berbagai platform yang telah ada sebelumnya, seperti iOS, *Blackberry* OS dan *Windows Phone* 7.

Keuntungan utama dari pengadopsian *Android* adalah sistem operasi ini menawarkan pendekatan seragam untuk pengembangan aplikasinya. Pengembang hanya perlu mengembangkan aplikasi *android* dan aplikasi mereka seharusnya dapat berjalan pada berbagai macam perangkat, selama alat tersebut memakai *android* sebagai sistem operasinya. Di dunia *smartphone*, aplikasi merupakan

bagian penting dari rantai kesuksesan. Produsen ponsel dengan itu melihat *android* sebagai harapan terbaik untuk menantang serangan Iphone, yang telah mengontrol sebagian besar berbasis aplikasi *mobile*.

II.3.1. Versi *Android*

a. *Android* versi 1.1

b. *Android* versi 1.5 (*Cupcake*)

c. *Android* versi 1.6 (*Donut*)

d. *Android* versi 2.0/2.1 (*Eclair*)

e. *Android* versi 2.2 (*Froyo*)

f. *Android* versi 2.3 (*Gingerbread*)

g. *Android* versi 3.0 (*Honeycomb*)

h. *Android* versi 4.0 (ICS: *Ice Cream Sandwich*)

i. *Android* versi 4.1 (*Jelly Bean*)

II.4. *Unified Modeling Language* (UML)

Unified Modeling Language adalah sekumpulan pemodelan konvensi yang digunakan untuk menentukan atau menggambarkan sebuah sistem perangkat lunak dalam kaitannya dengan objek. UML dapat juga diartikan sebuah bahasa grafik standar yang digunakan untuk memodelkan perangkan lunak berbasis objek. UML terdiri dari diagram-diagram, dimana setiap diagram dalam UML memperlihatkan sistem dari berbagai sudut pandang yang berbeda. Adapun definisi diagram-diagram tersebut adalah sebagai berikut:

1. *Use Case Diagram*

Use case diagram adalah sebuah gambaran dari fungsi sistem yang dipandang dari sudut pandang pemakai. *Actor* adalah segala sesuatu yang perlu berinteraksi dengan sistem untuk pertukaran informasi (Whitten, 2004: 258).

Merikut ini merupakan gambar dari tiga komponen sistem dalam *use case diagram*:

Tabel II.1. Komponen *Use Case Diagram*

Nama Komponen	Keterangan	Simbol
Use Case	*Use case* digambarkan sebagai lingkaran elips dengan nama *use case* dituliskan di dalam elips tersebut.	
Actor	*Actor* adalah pengguna sistem. *Actor* tidak terbatas hanya manusia saja, jika sebuah sitem berkomunikasi dengan aplikasi lain dan membutuhkan *input* atau memberika *output*, maka aplikasi tersebut juga bisa dianggap *actor*.	
Association	*Association* digunakan untuk menghubungkan *actor* dengan *use case*. Association digambarkan dengan sebuah garis yang menghubungkan antara *actor* dengan *use case*.	

(Sumber: Grady Booch; 1999)

2. *Sequence Diagram*

Sequence diagram menggambarkan interaksi *object* yang disusun dalam suatu urutan waktu. Diagram ini khusus berasosiasi dengan *use case*. *Sequence diagram* memperlihatkan tahap demi tahap apa saja yang seharusnya terjadi untuk menghasilkan sesuatu yang dilakukan dalam *use case*.

Adapun Komponen dalam *sequence diagram* adalah sebagai berikut:

Tabel II.2. Komponen *Sequence Diagram*

Nama Komponen	Keterangan	Simbol
Lifeline	*Lifeline* mengindikasikan keberadaan sebuah *object* dalam basis waktu. Notasi untuk *Lifeline* adalah garis putus-putus vertikal yang ditarik dari sebuah objek	
Activation	*Activation* dinotasikan sebagai sebuah kotak segiempat yang digambar pada sebuah *lifeline*. Mengindikasikan sebuah *object* yang akan melakukan sebuah aksi.	
Message	*Message* digambarkan dengan anak panah horizontal antara *activation*. Mengindikasikan komunikasi antara *object-object*.	

(Sumber: Grady Booch; 1999)

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

Adapun komponen dari *activity diagram* adalah sebagai berikut:

Tabel II.3. Komponen *Activity Diagram*

Simbol	Keterangan
	Titik awal
	Titik akhir
	*Activity*
	Pilihan untuk mengambil keputusan
	*Fork*, digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau menggabungkan dua kegiatan paralel menjadi satu.
	*Rake*, menunjukkan adanya dekomposisi.
	Tanda waktu
	Tanda pengiriman
	Tanda penerimaan
	Aliran akhir (*Flow Final*)

(Sumber: Grady Booch; 1999)