BAB II
LANDASAN TEORI

2.1. Rancang Bangun

Alvin (2006) mengungkapkan bahwa rancang bangun adalah suatu istilah umum untuk membuat atau mendesain suatu objek dari awal pembuatan sampai akhir pembuatan. Rancang bangun berawal dari kata desain yang artinya perancangan, rancang, desain, bangun sedangkan merancang artinya mengatur, cara, perbuatan merancang. Dapat disimpulkan arti kata desain adalah proses, cara, perbuatan dengan mengatur segala sesuatu sebelum bertindak atau merancang.

2.2. Aplikasi

Wardana (2010) mendefinisikan bahwa 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.

2.3. *Beatbox*

Menurut lederer (2006) mengatakan *Human Beatbox* adalah sebuah bentuk seni vokal yang menarik dan melibatkan manipulasi yang cepat dan tepat dari organ vokal untuk menirukan ketukan dari irama alat musik drum. Penelitian ini, diterbitkan sebagai rangkaian artikel, bertujuan untuk menilai akurasi suara

beatboxing, lalu meniru suara instrumen perkusi dan untuk menyelidiki bagaimana *beatboxing* diproduksi dan dirasakan dalam arti yang lebih luas dan dalam hubungannya dengan pengertian berbicara. Sebagai *interpolasi* suara *onomatope* dan efek vokal, *beatbox* juga membutuhkan tata bahasa yang lebih luas melalui rongga mulut untuk meniru suara mesin drum. Berbagai konsep mekanisme pelatihan dan metode artikulasi yang digunakan untuk membuat imitasi dalam penelitian ini bertujuan untuk menilai seberapa akurat metode ini yang memungkinkan *beatboxer* untuk mereproduksi suara mesin drum tiruan. Arti yang lebih luas dari *beatboxing* juga termasuk dalam integrasi dengan pola bernyanyi dan berbicara dari keseluruhan *beatboxing* oleh para pendengar.

2.4. *Android*

Masruri & Java (2015) mendefinisikan bahwa *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.

2.4.1 Fitur-fitur yang ada di *android*

Nazarudin (2014), menyatakan bahwa terdapat berbagai macam fitur dan versi dari *platform android*. Berikut penjelasan macam-macam fitur dan versi *android*:

a. *Framework* Aplikasi, pengembang memiliki akses penuh *framework* API (*Application Programming Interface*) yang sama yang digunakan oleh aplikasi inti.
b. Mesin *virtual Dalvik* dioptimalkan untuk perangkat *mobile*.
c. *Integrated browser* berdasarkan *engine open source WebKit*
d. Grafis yang dioptimalkan dan didukung oleh perpustakaan grafis 2D, grafis 3D berdasarkan spesifikasi *opengl* ES 1,0

e. *SQLite* untuk penyimpanan data
f. Media Support yang mendukung audio, video, dan gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
g. GSM *Telephony* (tergantung *hardware*)
h. *Bluetooth*, *EDGE*, 3G, dan *WiFi*
i. Kamera, GPS, kompas, dan *accelerometer*
j. Lingkungan *Development* yang lengkap: perangkat *emulator*, *tools* untuk *debugging*, profil dan kinerja memori, dan *plugin* untuk *Eclipse* IDE.

2.4.2 Versi *Android*

Masruri dan Java (2015) mengungkapkan bahwa keunikan dari sistem operasi *android* ini adalah dari tiap nama yang diberikan pada tiap versinya yang dimana 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 Buku Pintar *Android*.

a. *Android* Versi Beta

   Android versi beta ini dirilis pada tanggal 5 November 2007, dimana *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 *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* 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*, *codename* 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 *codename Gingerbread* 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*. Tak lama kemudian *Google* merilis versi terbarunya yaitu 3.1 dengan *codename* 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. *Android* versi ini 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. *Adobe Flash* CS6

Pramono (2012, p.2) juga mendefinisikan *Adobe Flash* CS6 adalah salah satu aplikasi pembuat animasi yang cukup dikenal saat ini. Berbagai fitur dan kemudahan yang dimiliki menyebabkan *Adobe Flash* CS6 menjadi program animasi favorit dan cukup populer. Tampilan *interface*, fungsi dan pilihan palet yang beragam, serta kumpulan *tool* yang sangat lengkap dan sangat membantu dalam pembuatan karya animasi yang menarik. *Adobe flash* seperti *software* campuran dimana didalamnya terdapat semua kelengkapan yang dibutuhkan. Mulai dari fitur menggambar, ilustrasi, mewarnai, animasi, dan *programming*. Kita dapat mendesain gambar atau objek yang akan kita animasikan langsung pada Flash. Fitur programming pada *Flash* menggunakan bahasa *ActionScript*. *ActionScript* dibutuhkan untuk memberi efek gerak dalam animasi. *ActionScript* di *flash* pada awalnya memang sulit dimengerti jika seseorang tidak mempunyai dasar atau mengenal *adobe flash*.

2.5.1 *Action Script* 3.0

Madcoms (2012) menjelaskan *Actionscript* sebagai bahasa pemrograman pada *adobe flash player* dan *adobe AIR*. Dimana bahasa pemrograman ini bisa interaktif, menangani data, dan banyak digunakan pada *Flash*, *Flex*, dan *AIR* baik konten maupun aplikasi. *Actionscript* dapat di eksekusi oleh *Actionscript Virtual Machine* (*AVM*), yang mana ini merupakan bagian dari *Flash* dan *AIR*. Bahasa pemrograman ini akan familiar dengan para *develover* dengan kemampuan yang minim terhadap *Object Oriented Programming*.

Actionscript 3.0 mempunyai kapabilitas yang lebih bagus dari pada *Actionscript* sebelumnya. Yaitu di desain untuk memfasilitasi kreasi atau pekerjaan yang mempunyai kompleksitas yang tinggi dalam membangun aplikasi dengan data set yang besar dan *Object Oriented*, *code bases* yang bisa dipakai berulang kali. Bahasa pemrograman ini dapat mengeksekusi 10 kali lebih cepat dari pada *actionscript* sebelumnya (Madcom, 2012).

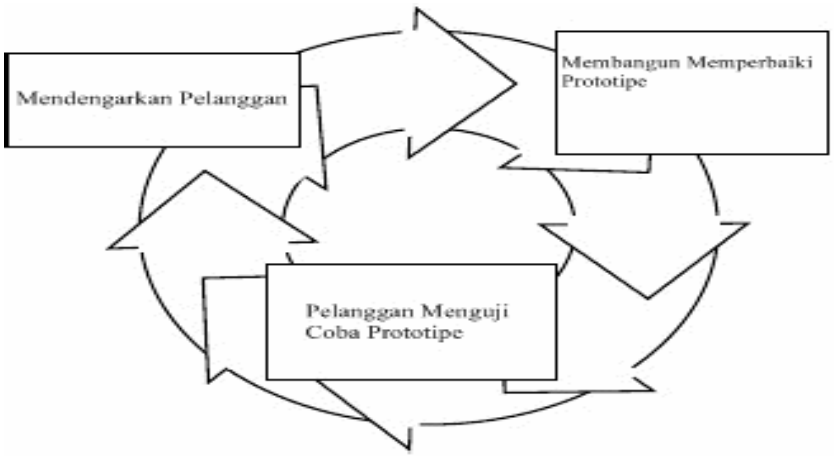
2.6. *Adobe AIR*

Wardana (2014) mengungkapkan bahwa *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*.

2.7. Metode *Prototype*

Sukamto dan Shalahuddin (2015) mendefinisikan bahwa model prototipe 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. Berikut ini adalah Gambar 2.1 dari model prototipe :

.



Gambar 2.1 Model Prototype

(Sumber : Sukamto dan Shalahuddin, 2015)

a) Pengumpulan kebutuhan Developer dan klien bertemu untuk menentukan tujuan umum, kebutuhan yang diketahui dan gambaran bagian-bagian yang akan dibutuhkan berikutnya. Detail kebutuhan mungkin tidak dibicarakan disini, pada awal pengumpulan kebutuhan.

Selanjutnya peneliti akan melakukanan analisis terhadap data apa saja yang dibutuhkan, seperti analisis terhadap sistem yang berjalan, analisis kebutuhan perangkat lunak, analisis kebutuhan perangkat keras, dan analisis kebutuhan materi pembelajaran.

b) Perancangan

Perancangan dilakukan dengan cepat dan rancangan mewakili semua aspek software yang diketahui, dan rancangan ini menjadi dasar pembuatan *prototype*. Dalam tahap ini peneliti akan membangun sebuah versi *prototype* yang dirancang kembali dimana masalah-masalah tersebut diselesaikan.

c) Evaluasi *prototype*

Pada tahap ini, calon pengguna mengevaluasi *prototype* yang dibuat dan digunakan untuk memperjelas kebutuhan *software*. *Software* yang sudah jadi dijalankan dan akan dilakukan perbaikan apabila kurang memuaskan. Perbaikantermasuk dalam memperbaiki kesalahan atau kerusakan yang tidak ditemukan pada langkah sebelumnya.

Menurut Sukamto dan Shalahuddin (2015) kelebihan *Prototype Model* adalah:

a) *End user* dapat berpartisi pasiaktif.
b) Penentuan kebutuhan lebih mudah diwujudkan.
c) Mempersingkat waktu pengembangan *software*.

Kekurangan *Prototype Model* adalah:

a) Prosesanalisis dan perancangan terlalu singkat.
b) Mengesampingkan alternatif pemecahan masalah.
c) Biasanya kurang fleksibel dalam menghadapi perubahan.
d) *Prototype* yang dihasilkan tidak selamanya mudah dirubah.
e) *Prototype* terlalu cepat selesai.

2.8 UML (*Unified Modeling Language*)

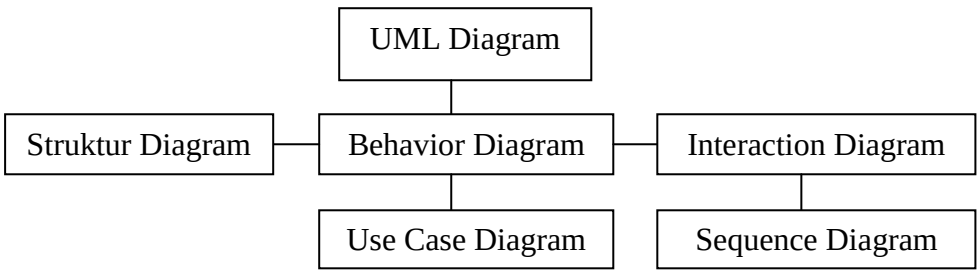
Sukamto dan Shalahuddin, (2015) *Unified Modelling Language* (UML) adalah salah satu standar bahasa yang banyak di gunakan di dunia industri untuk mendefenisikan *requirement*, membuat analisis & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek dengan menggunakan diagram dan teks-teks pendukung.

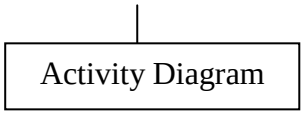
2.8.1 Diagram UML

Sukamto dan Shalahuddin, (2015) pada UML 2.3 terdiri dari 13 macam diagram yang di kelompokkan dalam 3 kategori.

1) *Structure diagrams* yaitu kumpulan diagram yang di gunakan untuk menggambarkan suatu struktur statis dari sistem yang di modelkan.
2) *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
3) *Interaction diagram* yaitu kumpulan diagram yang di gunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.

Lebih jelasnya dapat di lihat pada Gambar 2.2 berikut.





Gambar 2.2 Diagram UML

(Sukamto dan Shalahuddin, 2015)

Berikut penjelasan dari jenis-jenis diagram tersebut antara lain:

1) *Class* Diagram

   Menggambarkan struktur sistem dari segi pendefenisian kelas-kelas yang akan di buat untuk membangun sistem

2) *Object* Diagram

   Menggambarkan struktur sistem dari segi penanaman objek dan jalannya objek dalam sistem.

3) *Component* Diagram

   Menunjukan organisasi dan ketergantungan di antara kumpulan komponen dalam sebuah sistem.

4) *Composite Structure* Diagram

   Menggambarkan struktur dari bagian – bagian yang saling terhubung maupun mendeskripsikan struktur pada saat berjalan (*runtime*) dari *instance* yang saling terhubung.

5) *Package* Diagram

   Menyediakan cara mengumpulkan elemen-elemen yang saling terkait dalam diagram UML.

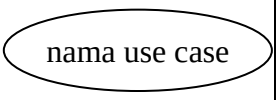
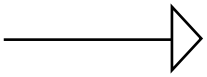
6) *Deployment* Diagram

   Menunjukan konfigurasi komponen dalam proses eksekusi aplikasi.

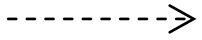
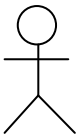
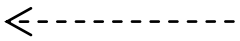
7) *Use Case* Diagram

   Pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan di buat dapat di lihat pada Tabel 2.1

Tabel 2.1. Simbol – simbol *use case diagram*

Nama	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.

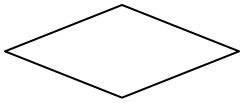
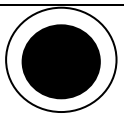
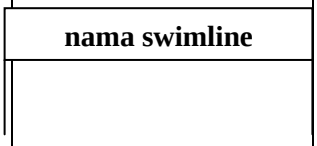
Tabel 2.1 Lanjutan Simbol *Use Case*

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.
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.
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.

8) *Activity* Diagram

Menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak dapat di lihat pada Tabel 2.2 berikut.

Tabel 2.2 Simbol-Simbol *Activity* Diagram

Nama	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.
Percabangan/*decision*		Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
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 bertanggungjawabterhadap aktivitas yang terjadi.

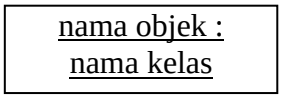
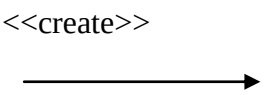
9) *State Machine* Diagram

Di gunakan untuk menggambarkan perubahan status atau transisi status dari sebuah mesin atau sistem atau objek.

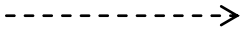
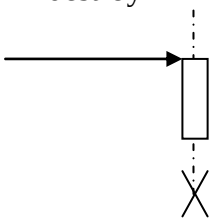
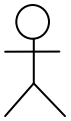
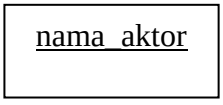
10) *Sequence* Diagram

Menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek dapat di lihat pada Tabel 2.3

Tabel 2.3 Simbol – simbol *sequence diagram*

Nama	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.

Tabel 2.3 Lanjutan Simbol – simbol *sequence diagram*

Pesan tipe call	1 : nama_metode() 	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri.
Pesan tipe send	1 : masukan 	Merupakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim.
Pesan tipe return	1 : keluaran 	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	<<destroy>> 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhirim sebaliknya jika ada create maka ada destroy.
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.