BAB II
LANDASAN TEORI

2.1 Rancang Bangun

(Alvin, 2006) 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. Rancang bangun berarti mengatur segala sesuatu (sebelum bertindak, megerjkan, atau melakukan sesuatu), merencanakan.

2.2 Aplikasi

(Wardana, 2010) 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*.

2.3 Sistem Informasi

(Yakub, 2012) Sistem informasi merupakan suatu komponen yang terdiri dari manusia, teknologi informasi dan prosedur kerja yang memproses, menyimpan, menganalisis dan menyebarkan informasi untuk mencapai suatu tujuan.

2.3.1 Komponen Sistem Informasi

(Aziz & Pujiono, 2006) terdapat beberapa komponen sistem informasi, yaitu :

1. *Input*

   Input adalah semua data yang diambil dan dikumpulkan untuk proses di dalam sistem informasi. Data yang diinputkan bisa dalam bentuk data analog maupun data digital.

2. Proses

   Proses merupakan kumpulan prosedur yang akan memanipulasi *input* yang kemudian akan disimpan dalam basis data dan selanjutnya akan diolah menjadi suatu *output* yang akan digunakan oleh *end user*. Manusia, perangkat komputer, prosedur dan penyimpanan data adalah empat sumber utama dalam proses sistem informasi.

3. *Output*

   Output merupakan semua keluaran dari model yang sudah diolah menjadi suatu informasi yang berguna dan dapat dipakai oleh penerima.

4. Teknologi

   Teknologi berfungsi untuk memasukan, mengolah dan menghasilkan keluaran.

5. Kontrol

   Kontrol merupakan semua tindakan yang diambil untuk menjaga sistem berjalan menuju tujuannya.

2.4 Sistem *Vertikultur*

(L & Saparinto, 2016) *Vertikultur* diambil dari istilah *verticulture* dalam bahasa Inggris (*vertical* dan *culture*), artinya sistem budidaya pertanian yang dilakukan secara *vertical* atau bertingkat. Cara bercocok tanam secara *vertikultur* ini sebenarnya sama saja dengan bercocok tanam di kebun atau di sawah. Perbedaanya terletak pada lahan yang digunakan dan wadah untuk menanam.

Banyak sedikitnya tanaman yang kita budidayakan bergantung pada model wadah yang akan digunakan. Model, bahan, ukuran, wadah *vertikultur* sangat banyak, tinggal disesuaikan dengan kondisi dan keinginan. Pada umumnya adalah berbentuk persegi panjang, segi tiga, atau dibentuk mirip anak tangga, dengan beberapa undak-undakan atau sejumlah rak. Bahan dapat berupa bambu atau pipa paralon, kaleng bekas, botol plastik, bahkan lembaran karung beras pun bisa, karena salah satu filosofi dari *vertikultur* adalah memanfaatkan benda-benda bekas di sekitar kita.

2.4.1 Kelebihan Sistem *Vertikultur*

(L & Saparinto, 2016) Adapun kelebihan dari bercocok tanam dengan sistem *verikultur* adalah sebagai berikut :

a. Efesiensi dalam penggunaan lahan dan air.
b. Wadah tanaman yang mudah didapat karena menggunakan wadah-wadah bekas atau tidak terpakai.
c. Penghematan pemakaian pupuk dan pestisida.
d. Dapat dipindahkan dengan mudah karena tanaman diletakkan dalam wadah tertentu.
e. Mudah dalam hal monitoring atau perawatan tanaman.
f. Menghasilakan tanaman organik yang sehat.
g. Umur tanaman relativ pendek.
h. Menghindarkan dari jangkauan hewan ternak.

2.4.2 Kekurangan Sistem *Vertikultur*

(L & Saparinto, 2016) Adapun kekurangan bercocok tanam dengan sistem *vertikultur* adalah sebagai berikut :

a. Sistem penyiraman harus *continue* serta memerlukan beberapa peralatan tambahan, misalnya tangga.

b. Hanya dapat dikembangkan untuk jenis tanaman yang mempunyai nilai ekonomis tinggi, berumur panen pendek, dan berakar pendek.

2.4.3 Jenis-jenis Tanaman Sistem *Vertikultur*

(L & Saparinto, 2016) Jenis tanaman yang dapat ditanam dengan sistem *vertikultur* ini sangat banyak seperti :

a. Tanaman sayur semusim (sawi, selada, kangkung, kubis, wortel, terong, cabai, tomat, bayam dan lain-lain)
b. Tanaman bunga seperti anggrek, mawar, melati, kembang sepatu dan lain-lain.
c. Tanaman obat-obatan yang sekulen.

2.4.4 Aspek Sistem Vertikultur

(L & Saparinto, 2016) Terdapat beberapa aspek yang harus dipersiapkan dalam bercocok tanam dengan sistem *vertikultur* diantaranya :

1. Pembuatan Wadah dan Rak *Vertikultur*

   Wadah tanam dalam sistem *vertikultur* biasanya menggunakan wadah-wadah bekas yang ada disekitar kita. Seperti paralon, bambu, pot, polibag, botol plastik bekas dan lain-lain. Rak pada sistem vertikultur disesuaikan dengan kebutuhan. Biasanya dapat menempel di dinding, menggantung, atau dengan tiang penyangga.

2. Penyiapan Media Tanam

   Media tanam adalah tempat tumbuhnya tanaman untuk menunjang perakaran. Dari media tanam inilah tanaman menyerap makanan berupa unsur hara melalui akarnya. Adapun media tanam berupa tanah, pupuk kandang, sekam, dan pasir dengan perbandingan 1:1:1:1. Sekam berfungsi untuk menampung air di dalam tanah sedangkan kompos menjamin tersedianya bahan penting yang akan diuraikan menjadi unsur hara yang diperlukan tanaman. Fungsi pupuk kandang adalah sebagai

sumber makanan bagi tanaman dan sebagai pengikat air, agar media tidak cepat kering. Pasir berfungsi untuk mempermudah mengalirnya kelebihan air dalam media tanam dan mengurangi mengerasnya media tanam.

3. Penanaman Tanaman

   Pada dasarnya ada tiga tahap dalam aspek ini, yaitu persemaian, pemindahan, dan penanaman.

   a. Penyemaian benih

      Seperti halnya menanam, menyemaikan benih juga memerlukan wadah dan media tanam. Wadah diisi media tanam seperlunya dan memiliki lubang di bagian bawah untuk mengeluarkan air. Jumlah benih yang dapat disemai disesuaikan dengan ukuran wadahnya, dalam hal ini jarak tanam benih diatur sedemikian rupa agar tidak berdempetan.

   b. Pemindahan dan penanaman tanaman

      Bibit tanaman yang dipindahkan ke wadah *vertikultur* sudah berumur lebih dari satu bulan. Idealnya benih yang tumbuh daun berjumlah 4-5 helai.

4. Perawatan Tanaman

   Tanaman juga memerlukan perawatan, seperti halnya makhluk hidup yang lain. Adapun perawatan tanaman yang harus dilakukan adalah sebagai berikut:

   a. Penyiraman

      Penyiraman dilakukan setiap hari, hal ini bertujuan agar tanaman tidak kekurangan kandungan air. Jika musim kemarau penyiraman dilakukan lebih sering dari pada musim penghujan.

   b. Pemupukan

      Sebaiknya pupuk yang digunakan adalah pupuk organik misalnya pupuk kompos, pupuk kandang atau pupuk bokashi.

c. Pengendalian Hama dan penyakit

Pengendalian hama dan penyakit sebenarnya tidak harus menggunakan pestisida atau bahan kimia lainya. Obat semprot yang alami dapat dibuat sendiri dengan memanfaatkan sisa sayuran yang tidak dimasak dibusukkan dengan air bekas cucian beras lalu dicampur krim sabun cuci sedikit atau secukupnya.

5. Pemanenan Tanaman

Pemanenan tanaman biasanya dilakukan dengan sistem cabut akar atau potong untuk sayuran seperti sawi, selada, kangkung dan sebagainya. Apabila tanaman hanya untuk dikonsumsi sendiri akan lebih menghemat apabila panen dilakukan dengan mengambil daunnya saja. Cara tersebut tanaman bisa bertahan lebih lama dan bisa dipanen berulang-ulang. Tanaman yang akan dipanen harus diseleksi dahulu agar tepat. Sayuran bisa mulai dipanen pada umur 40-45 hari pasca tanam. Selain itu, panen ditujukan sesuai kebutuhan pasar, terutama standar dan ukurannya. Sebaiknya, yang dipanen terlebih dahulu adalah sayuran berukuran besar.

6. Pasca Panen

Setelah dipanen, sayuran harus diperlakukan sebaik mungkin. Tujuannya agar produk sayur tidak rusak dan tetap dalam kondisi segar serta layak konsumsi atau jual. Untuk sayuran yang sudah dipanen, sebaiknya segera dicuci bersih, lalu dipilih yang tidak rusak. Untuk sayuran hasil panen yang akan dijual dapat dikemas dengan kemasan plastik khusus sehingga memungkinkan untuk memiliki nilai jual yang lebih tinggi dan bisa dijual dipasar modern, tetapi juga bisa dikemas secara sederhana seperti diikat secara rapi dan teratur.

(Soeleman & Rahayu, 2013) Setelah panen dari wadah seperti pot, botol palstik dan sebagainya, keluarkan media tanam dan bersihkan sisa akar. Jemur media tanam agar hama dan penyakit mati terpapar sinar

matahari. Tambahkan pupuk organik atau pupuk kandang dan aduk sambil digemburkan, media tanam siap digunakan kembali. Untuk wadah tanam seperti botol plastik, paralon, ember, bambu dapat digunakan kembali secara berkali-kali setelah panen. Tetapi untuk wadah tanam seperti *polibag* biasanya hanya dapat digunakan satu kali saja setelah panen, hal ini dikarenakan wadah tanam seperti *polibag* cepat rapuh atau rusak.

2.5 *Android*

(M. Hilmi Masruri & 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 Kelebihan *Android*

(Ed Burnette, 2010) dalam bukunya *Hello Android, Third Edition, android* memiliki keunggulan antara lain :

1. 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.
2. 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.*

3. 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*.
4. 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.

2.6 *Adobe Flash Profesional CS6*

(Madcoms, 2012) *Adobe Flash Profesional CS6* adalah salah satu perangkat lunak komputer yang merupakan produk unggulan *Adobe System*. *Adobe Flash Profesional CS6* merupakan *software* yang digunakan untuk menciptakan animasi dan konten multimedia. Dengan *Adobe Flash Profesional CS6* kita dapat dengan mudah menggabungkan beberapa simbol dan urutan animasi menjadi lembaran *sprite* tunggal dan dioptimalkan untuk alur kerja yang lebih baik, dibuat lebih menarik dengan konten menggunakan ekstensi asli untuk mengakses kemampuan perangkat secara spesifik. *Adobe Flash Profesional CS6* telah membuktikan dirinya sebagai program animasi dua dimensi berbasis *vector* dengan kemampuan profesional. Dalam perkembangannya, *Adobe Flash* selalu melakukan banyak penyempurnaan pada setiap versinya. *Adobe Flash Profesional CS6* menghadirkan fitur-fitur baru yang menjadikan *flash* semakin diakui sebagai program yang handal.

2.6.1 Kelebihan *Adobe Flash*

Adapun kelebihan dari *Adobe Flash* adalah sebagai berikut :

1. merupakan teknologi animasi web yang paling populer hingga saat ini sehingga banyak pihak yang mendukung teknologi ini.
2. Ukuran file yang kecil dengan kualitas yang baik.
3. Kebutuhan hardware yang tidak tinggi.
4. Dapat membuat *website*, cd-interaktif, animasi web, animasi kartun, kartu elektronik, iklan TV, banner di web, presentasi interaksi, permainan, aplikasi web dan *handphone*.
5. Dapat ditampilkan di berbagai media seperti Web, CD-ROM, VCD, DVD, Televisi, *Handphone* dan PDA.
6. Adanya *Actionscript*. Dengan *actionscript* kita dapat membuat animasi dengan menggunakan kode sehingga memperkecil ukuran *file*. Karena adanya *actionscript* ini juga *Flash* dapat untuk membuat *game* karena *script* dapat menyimpan *variable* dan nilai, melakukan perhitungan, dsb. yang berguna dalam *game*. Selain itu, *Flash* adalah program berbasis vektor.
7. Hasil akhir dapat disimpan dalam berbagai macam bentuk seperti *.avi, *.gif, *.mov, maupun file dengan format.

2.6.2 Kekurangan *Adobe Flash*

Selain memiliki beberapa kelebihan, *Adobe Flash* juga memiliki beberapa kekurangan diantaranya adalah sebagai berikut :

1. Salah satunya adalah komputer yang ingin memainkan animasi *flash* harus memiliki *flash player*. Anda harus menginstallnya, biasanya secara *online*.
2. Program *adobe flash* bukan *freeware*.
3. Grafisnya kurang lengkap.
4. Menunya tidak user *friendly.*

5. Bahasa pemrogramannya agak sedikit susah.
6. Kurang dalam animasi 3D.
7. Belum ada *template* di dalamnya.

2.7 *Action Script 3.0*

(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*.

2.8 *Adobe AIR*

(Komputer, 2014) *Adobe AIR* adalah sebuah *cross operation system runtime* yang di kembangkan oleh *Adobe* sehingga memungkinkan pengembang memanfaatkan keterampilan seperti (*Flash, Flex, HTML, JavaScript*, dan *PDF*) untuk membuat RIA (*Rich Internet Application*) dan kontennya ke dalam *platform* baru. Berbeda dengan aplikasi web pada umumnya, aplikasi AIR ini di-install pada desktop dan bisa beroperasi secara *offline*. Karakteristiknya hampir sama dengan aplikasi desktop biasa. Fungsi dari *Adobe AIR* ialah sebagai media antar muka yang bekerja pada berbagai *platform* yang berjalan pada sistem operasi *Windows*, *Linux* maupun *MacOS* seperti *Andriod*, *Iphone* dan sebagainya. Aplikasi AIR memiliki kemampuan untuk mengakses data yang tersimpan pada komputer lokal. Jadi dengan kata lain AIR bisa menyimpan, membuka, dan mengedit data

atau informasi pada komputer *client*. Ini yang membedakannya dengan aplikasi web biasa yang berjalan di *browser*.

2.9 *Unified Modeling Language* (UML)

(Haviluddin, 2011) dalam Jurnal Informatika *Memahami Penggunaan UML* mengatakan bahwa *Unified Modelling Language* (UML) adalah suatu alat untuk memvisualisasikan dan mendokumentasikan hasil analisa dan desain yang berisi sintak dalam memodelkan sistem secara visual. (S, Rosa A. dan M. Shalahuddin, 2016) *Unified Modeling language* atau UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks pendukung. UML hanya berfungsi untuk melakukan pemodelan. 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.

2.9.1 Diagram UML

(S, Rosa & M, 2016) UML terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori. Berikut merupakan penjelasan tentang pembagian kategori tersebut :

a. *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
b. *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
c. *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.

2.9.1.1 *Class* Diagram

(S, Rosa & M, 2016) 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. Atribut merupakan *variable-variable* yang dimiliki oleh suatu kelas, dan operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas. 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 :

a. Kelas main

   Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.

b. Kelas yang menangani tampilan sistem (*view*)

   Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.

c. 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.

d. 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.

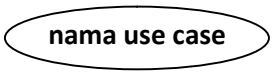
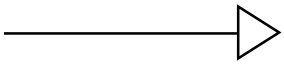
2.9.1.2 *Use Case* Diagram

(S, Rosa & M, 2016) 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.*

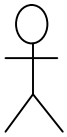
a. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
b. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

(S, Rosa & M, 2016) Berikut merupakan simbol-simbol pada diagram *use case* :

Tabel 2.1 Simbol Diagram *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.

Lanjutan **Tabel 2.1** Simbol Diagram *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.

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.

2.9.1.3 *Activity* Diagram

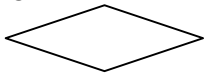
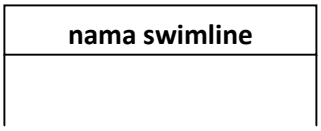
(S, Rosa & M, 2016) *Activity* diagram menggambarkan *workflow* (alir kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah

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.

(S, Rosa & M, 2016) Berikut merupakan simbol-simbol *activity* diagram :

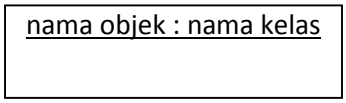
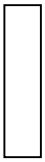
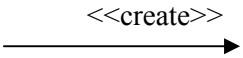
Tabel 2.2 Simbol *Activity* Diagram

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.
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 bertanggungjawab terhadap aktivitas yang terjadi.

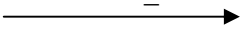
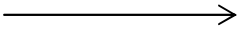
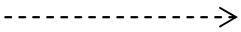
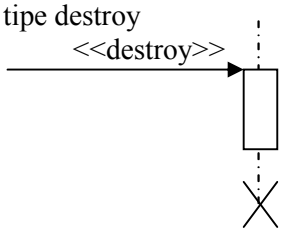
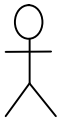
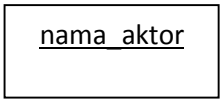
2.9.1.4 *Sequence* Diagram

(S, Rosa & M, 2016) *Sequence* Diagram 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 *sequence* 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. Berikut merupakan simbol-simbol *sequence* diagram :

Tabel 2.3 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.

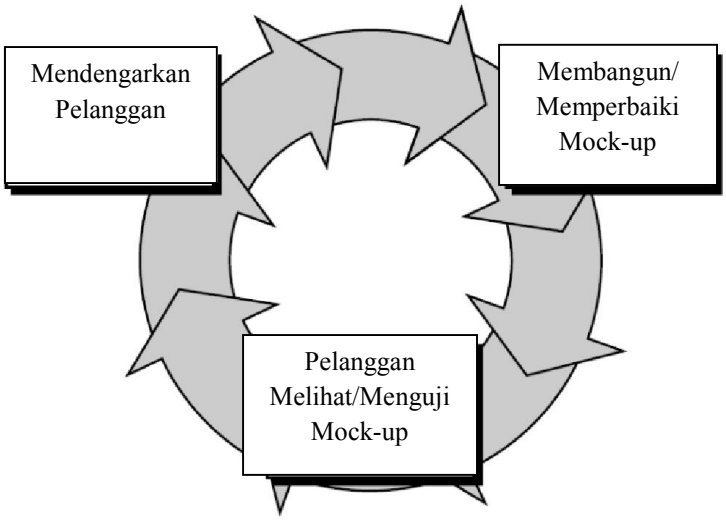
Lanjutan **Tabel 2.3** Simbol *Sequence* Diagram

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 Atau orang   	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.

2.10 Model *Prototype*

(S, Rosa & M, 2016) Model *prototype* (*prototyping model*) dimulai dari mengumpulkan kebutuhan pelanggan terhadap perangkat lunak yang akan dibuat. Lalu dibuatlah program *prototype* agar pelanggan lebih terbayang dengan apa yang sebenernya diinginkan. Program *prototype* 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 *prototype* ini dievalusai oleh pelanggan atau *user* sampai ditemukan spesifikasi yang sesuai dengan keinginan pelanggan atau *user*. (S, Rosa & M, 2016) Berikut adalah gambar dari model *prototype* :



Gambar 2.1 Model Prototype

(S, Rosa & M, 2016) *Mock-up* adalah sesuatu yang digunakan sebagai model desain yang digunakan untuk mengajar, demonstrasi, evaluasi desain, promosi, atau keperluan lain. Sebuah *mock-up* disebut sebagai *prototype* perangkat lunak jika menyediakan atau mampu mendemonstrasikan sebagian besar fungsi sistem perangkat lunak dan memungkinkan pengujian desain sistem perangkat lunak. Iterasi terjadi pada pembuatan *prototype* sampai sesuai dengan keinginan

pelanggan (*customer*) atau *user*. Proses pada model *prototyping* dapat dijelaskan sebagai berikut :

a. Pengumpulan kebutuhan : *Developer* dan klien bertemu dan menentukan tujuan umum, kebutuhan yang diketahui dan gambaran bagian-bagian yang akan dibutuhkan berikutnya.
b. Perancangan : Perancangan dilakukan cepat dan rancangan mewakili semua aspek perangkat lunak yang diketahui, dan rancangn ini menjadi dasar pembuatan *prototype.*
c. Evaluasi *prototype* : Klien mengevaluasi *prototype* yang dibuat dan digunakan untuk memperjelas kebutuhan perangkat lunak. Perulangan ketiga proses ini terus berlangsung hingga semua kebutuhan terpenuhi. *Prototype-prototype* dibuat untuk memuaskan kebutuhan klien dan untuk membangun perangkat lunak lebih cepat, namun tidak semua *prototype* bisa dimanfaatkan.

2.11 Populasi dan Sampel

(Natasari, Sahfitri, Kom, & Kom, 2016) dalam Jurnal Teknik Informatika *Evaluasi Investasi Teknologi Informasi Menggunkan ISO/IEC,* Populasi adalah keseluruhan objek penelitian. Poulasi yaitu sekelompok orang, kejadian, atau segala yang mempunyai karakteristik tertentu. Sampel adalah sebagian dari jumlah dan karakteristik yang dimiliki oleh populasi tersebut atau bagaian kecil dari anggota populasi yang diambil menurut prosedur tertentu sehingga dapat mewakili populasinya.

Secara umum, ada dua jenis teknik pengambilan sampel yaitu :

2.11.1 *Random Sampling (Probability Sampling)*

Random sampling yaitu suatu teknik sampling yang memberikan peluang atau kesempatan yang sama bagi setiap unsur (anggota) populasi untuk dipilih menjadi anggota sampel. Teknik ini terdiri atas :

a. *Simple Random Sampling*

Pengambilan sampel anggota populasi dilakukan secara acak, tanpa memperhatikan strata yang terdapat dalam populasi tersebut.

b. *Dispropotionate Stratified Random*

Suatu teknik yang digunakan untuk menentukan jumlah sampel, jika populasi berstrata tetapi kurang proposional.

c. *Propotionate Stratified Random*

Salah satu teknik yang digunakan jika populasi mempunyai anggota yang tidak homogen serta berstrata secara proposional.

d. Area *Sampling*

Dipakai untuk menentukan sampel jika objek yang akan diteliti atau sumber data sangat luas.

2.11.2 *Nonrandom Sampling (Nonprobability Sampling)*

Nonrandom sampling yaitu teknik yang tidak memberikan peluang atau kesempatan sama bagi setiap unsur atau anggota populasi dipilih menjadi sampel. Teknik ini terdiri atas :

a. *Sampling* Sistematis

Suatu teknik pangambilan sampel berdasarkan urutan dari anggota populasi yang telah diberi nomor urut.

b. *Sampling* Kuota

Teknik untuk menentukan sampel yang berasal dari populasi yang memiliki ciri-ciri tertentu sampai jumlah kuota yang diinginkan.

c. *Sampling* Aksidental

Teknik penentuan sampel berdasarkan kebetulan, yaitu siapa saja yang secara kebetulan bertemu dengan peneliti dapat dipakai sebagai sampel jika dipandang cocok sebagai sumber data.

2.12 Skala *Likert*

(Much, Subroto, Farisa, & Haviana, 2016) dalam Jurnal Transistor Elektro dan Informatika, Skala *likert* adalah skala pengukuran yang dikembangkan oleh *Likert*. Skala *likert* mempunyai empat atau lebih butir-butir pertanyaan yang dikombinasikan sehingga membentuk sebuah skor atau nilai yang mempresentasikan sifat individu. Skala *likert* adalah suatu skala psikometrik yang umum digunakan dalam perhitungan kuesioner yang dibagikan kepada responden untuk mengetahui skala sikap suatu objek tertentu.

(Sugiyono, 2012) Kuesioner merupakan tehnik pengumpulan data yang dilakukan dengan cara memberi seperangkat pertanyaan atau pernyataan tertulis kepada responden untuk dijawab.

Biasanya dalam skala *likert* disediakan lima pilihan dengan format seperti Sangat Setuju (SS), Setuju (S), Netral (N), Tidak Setuju (TS), Sangat Tidak Setuju (STS).

2.12.1 Penentuan Skor Jawaban

Skor jawaban merupakan nilai yang akan diberikan oleh responden. (Sugiyono, 2012) hal pertama yang harus kita lakukan adalah menentukan skor dari tiap jawaban yang akan diberikan misal :

a. Sangat Setuju (SS) diberi skor 5
b. Setuju (S) diberi skor 4
c. Netral (N) diberi skor 3
d. Tidak Setuju (TS) diberi skor 2
e. Sangat Tidak Setuju (STS) diberi skor 1

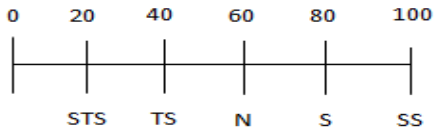
2.12.2 Skor Ideal

Merupakan skor yang digunakan untuk menghitung skor dalam menentukan rating scaledan jumlah seluruh jawaban. (Sugiyono, 2012) Untuk menghitung jumlah skor ideal digunakan rumus berikut:

Skor kriterium = Nilai skala x jumlah responden

2.12.3 Rating Scale

Skor yang telah diperoleh dari perhitungan skor ideal dimasukkan ke dalam *rating scale* seperti berikut :



Gambar 2.2 Rating Scale

Rating scale berfungsi untuk mengetahui hasil dari data kuesioner secara umum dan keseluruhan yang didapat dari penilaian kuesioner.

2.12.4 Persentase Persetujuan

(Sugiyono, 2012) Untuk mengetahui jumlah jawaban dari para responden melalui persentase yaitu digunakan rumus berikut:

$$p = \frac{f}{n} x\ 100\%$$

Gambar 2.3 Rumus Persentase Persetujuan

Keterangan :

p : persentese

f : frekuensi dari setiap jawaban kuesioner

n : jumlah skor ideal