BAB II

TINJAUAN PUSTAKA

II.1. Perancangan

Perancangan mempunyai 2 maksud, yaitu untuk memenuhi kebutuhan kepada pemakai sistem dan untuk memberikan gambaran yang jelas kepada pemogram komputer dan ahli-ahli teknik lainnya yang terlibat. (*Hanik Mujiati ; 2014 : 2*)

II.2. Keamanan

Masalah keamanan komputer merupakan sesuatu yang sangat penting dalam era informasi ini terutama bagi suatu organisasi atau perusahaan. Kerahasiaan data merupakan sesuatu yang penting dalam keamanan data. Data pelanggan menjadi salah satu data yang sangat penting dalam kelangsungan berjalannya perusahaan. (*Putu H. Arjana ; 2012 : 164*)

Keamanan merupakan bentuk tindakan untuk mempertahankan sesuatu hal dari berbagai macam gangguan dan ancaman. Aspek yang berkaitan dengan suatu keamanan dalam dunia komputer, antara lain:

1. *Privacy*/*Confidentiality* yaitu usaha menjaga
2. informasi dari orang yang tidak berhak mengakses (menggaransi bahwa data pribadi tetap pribadi).
3. *Integrity* yaitu usaha untuk menjaga data atau sistem tidak diubah oleh yang tidak berhak.

4. *Authentication* yaitu usaha atau metoda untuk mengetahui keaslian dari informasi yang dikirim dibuka oleh orang yang benar (asli).
5. *Availability* berhubungan dengan ketersediaan sistem dan data (informasi) ketika dibutuhkan. (*Putu H. Arjana ; 2012 : 164*)

Keamanan data ini merupakan salah satu aspek yang sangat penting dalam penggunaan komputer. Pemilik data tersebut tentunya ingin datanya aman terhadap gangguan dari berbagai tindakan yang tidak di inginkan, baik dari komputer pribadi (PC) ataupun jaringan. (*Putu H. Arjana ; 2012 : 164*)

II.3. Aplikasi

Aplikasi adalah suatu subkelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yagn diinginkan pengguna. Contoh utama aplikasi adalah pengolah kata, lembar kerja, memanipulasi foto, merancang rumah dan pemutar media. Beberapa aplikasi yang digabung bersama menjadi sutau pake disebut sebagai suatu paket atau suite aplikasi (*application suite*). Contohnya adalah *Microsoft Office* dan *OpenOffice.org*, yang menggabungkan suatu aplikasi pengolah kata, lembar kerja dan beberapa aplikasi lainnya. (*Dahlan Abdullah ; 2013 : 152*)

II.4. Pembelajaran

Pembelajaran adalah proses interaksi peserta didik dengan pendidik dan sumber belajar pada suatu lingkungan belajar. Pembelajaran merupakan bantuan yang diberikan pendidik agar dapat terjadi proses perolehan ilmu dan pengetahuan, penguasaan kemahiran dan tabiat, serta pembentukan sikap

dan kepercayaan pada peserta didik. Dengan kata lain, pembelajaran adalah proses untuk membantu peserta didik agar dapat belajar dengan baik. (*M. Rosidi Zamroni ; 2013 : 490*)

Pembelajaran yang dibantu oleh komputer dikenal dengan nama CAI yaitu ”*Computer Assisted Instruction*”. Prinsip pembelajaran ini menggunakan komputer sebagai alat bantu penyampaian pelajaran kepada user secara interaktif. (*Arif Aliyanto ; 20122 : E-30*)

II.5. Kriptografi

Kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan (*Cryptography is the art and science of keeping messages secure*). Kata “seni” di dalam definisi di atas berasal dari fakta sejarah bahwa pada masa-masa awal sejarah kriptografi, setiap orang mungkin mempunyai cara yang unik untuk merahasiakan pesan. Cara-cara unik tersebut mungkin berbeda-beda pada setiap pelaku kriptografi sehingga setiap cara menulis pesan rahasia pesan mempunyai nilai estetika tersendiri. Pada perkembangan selanjutnya, kriptografi berkembang menjadi sebuah disiplin ilmu sendiri karena teknik-teknik kriptografi dapat diformulasikan secara matematik sehingga menjadi sebuah metode yang formal. (*Putu H. Arjana ; 2012 : 164*)

II.5.1. Istilah-Istilah Dalam Kriptografi

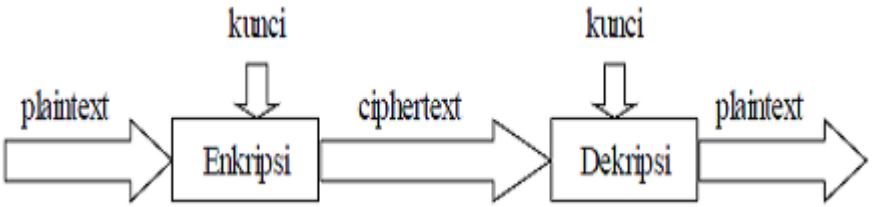
Berikut adalah istilah-istilah yang digunakan dalam bidang kriptografi :

1. *Plaintext* (M) adalah pesan yang hendak dikirimkan (berisi data asli).
2. *Ciphertext* (C) adalah pesan ter-enkrip (tersandi) yang merupakan hasil

enkripsi.

3. Enkripsi (fungsi E) adalah proses pengubahan *plaintext* menjadi *ciphertext*.
4. Dekripsi (fungsi D) adalah kebalikan dari enkripsi yakni mengubah *ciphertext* menjadi *plaintext*, sehingga berupa data awal/asli.
5. Kunci adalah suatu bilangan yang dirahasiakan yang digunakan dalam proses enkripsi dan dekripsi. (*Putu H. Arjana ; 2012 : 165*)

Kriptografi itu sendiri terdiri dari dua proses utama yakni proses enkripsi dan proses dekripsi. Seperti yang telah dijelaskan di atas, proses enkripsi mengubah plaintextmenjadi *ciphertext* (dengan menggunakan kunci tertentu) sehingga isi informasi pada pesan tersebut sukar dimengerti. Untuk melihat ilustrasi dari proses kriptografi dapat dilihat pada gambar II.1, mekanisme kriptografi. (*Putu H. Arjana ; 2012 : 165*)



Gambar II.1. Mekanisme Kriptografi
Sumber : (*Putu H. Arjana ; 2012 : 165*)

II.5.2. Kriptografi Kunci Simetri (Kriptografi Kunci-Privat)

Pada sistem kriptografi kunci-simetri, kunci untuk enkripsi sama dengan kunci untuk dekripsi, oleh karena itulah dinamakan kriptografi simetri. Keamanan sistem kriptografi simetri terletak pada kerahasiaan kuncinya. Ada banyak algoritma kriptografi modern yang termasuk ke dalam sistem kriptografi simetri, diantaranya adalah DES (*Data Encryption Standard*), Blowfish, Twofish, Triple-

DES, IDEA, Serpent, AES (*Advanced Encryption Standard*). (*Suriski Sitinjak ; 2010 : C-79*)

Algoritma kriptografi (*cipher*) simetri dapat dikelompokkan menjadi dua kategori, yaitu (*Suriski Sitinjak ; 2010 : C-79*) :

1. Cipher aliran (*stream cipher*)

   Algoritma kriptografi beroperasi pada plainteks/cipherteks dalam bentuk bit tunggal, yang dalam hal ini rangkaian bit dienkripsikan/didekripsikan bit per bit.

2. Cipher blok (*block cipher*)

   Algoritma kriptografi beroperasi pada plainteks/cipherteks dalam bentuk blok *bit*, yang dalam hal ini rangkaian *bit* dibagi menjadi blok-blok *bit* yang panjangnya sudah ditentukan sebelumnya.

II.5.3. Algoritma kriptografi

Berdasarkan kunci yang digunakan algoritma kriptografi terbagi atas dua bagian, yaitu :

1. Kriptografi *Secret Key*

   Kriptografi *secret key* adalah kriptografi yang hanya melibatkan satu kunci dalam proses enkripsi dan dekripsi. Proses dekripsi dalam kriptografi *secret key* ini adalah kebalikan dari proses enkripsi. Kriptografi simetris dapat dibagi menjadi dua, yaitu penyandian blok dan penyandian alir. Penyandian blok bekerja pada suatu data yang terkelompok menjadi blok-blok data atau kelompok data dengan panjang data yang telah ditentukan. Pada penyandian blok, data yang masuk akan dipecah-pecah menjadi blok data yang telah

ditentukan ukurannya. Penyandian alir bekerja pada suatu data *bit* tunggal atau terkadang dalam satu *byte*. Jadi format data yang mengalami proses enkripsi dan dekripsi adalah berupa aliran *bit-bit* data. (*Busran ; 2012 : 35*)

Contoh-contoh algoritma yang menggunakan kunci simetri yaitu :

- Data *Encryption Standarad* (DES)
- *Advanced Encryption Standard* (AES)
- *International* Data *Encryption Algorithm* (IDEA)
- *One Time Pad* (OTP)
- A5
- RC2
- RC4
- RC5
- RC6
- *Tiny Encryption Algorithm* (TEA)
- *Twofish*

2. Kriptografi *Public Key*

Kriptografi *public key* sering disebut dengan kriptografi asimetris. Berbeda dengan kriptografi *secret key*, kunci yang digunakan pada proses enkripsi dan proses dekripsi pada kriptografi *public key* ini berbeda satu sama lain. Jadi dalam kriptografi *public key*, suatu *key* generator akan menghasilkan dua kunci berbeda dimana satu kunci digunakan untuk melakukan proses enkripsi dan kunci yang lain digunakan untuk melakukan proses dekripsi. Kunci yang digunakan untuk melakukan enkripsi akan

dipublikasikan kepada umum untuk dipergunakan secara bebas. Oleh sebab itu, kunci yang digunakan untuk melakukan enkripsi disebut juga sebagai *public key*. Sedangkan kunci yang digunakan untukmelakukan dekripsi akan disimpan oleh pembuat kunci dan tidak akan dipublikasikan kepada umum. Kunci untuk melakukan dekripsi ini disebut *private key*. Dengan cara demikian, semua orang yang akan mengirimkan pesan kepada pembuat kunci dapat melakukan proses enkripsi terhadap pesan tersebut, sedangkan proses dekripsi hanya dapat dilakukan oleh pembuat atau pemilik kunci dekripsi. (*Busran ; 2012 : 36*) Yang termasuk algoritma asimetri adalah :

- *Digital Signature Algorithm* (DSA)
- *Ron Rivest, Adi Shamir* dan *Leonard Adleman* (RSA)
- *Diffie Helman* (DH)
- *Elliptic Curve Cryptography* (ECC)
- *Quantum*

II.6. Algoritma RC4

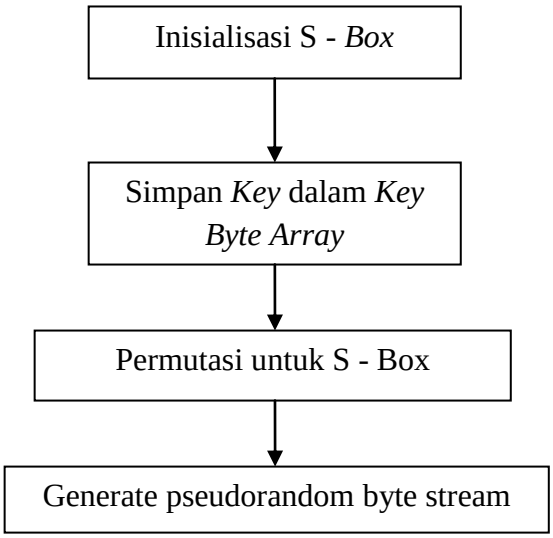
Kriptografi terdiri atas kunci simetri dan asimetri. Algoritma kunci simetri termasuk algoritma yang masih sering digunakan dalam pembuatan algoritma kriptografi. Bila dibandingkan dengan block cipher, maka stream cipher memiliki algoritma yang lebih sederhana. Salah satunya adalah RC4, yaitu algoritma yang masih banyak digunakan karena kesederhanaannya secara matematis dan kecepatan prosesnya yang tidak kalah cepat dibandingkan dengan algoritma yang lebih rumit. (*Busran ; 2012 : 33*)

RC4 merupakan jenis stream cipher yang mempunyai sebuah SBox,

S0,S1,...,S255, yang berisi permutasi dari bilangan 0 sampai 255, dan permutasi merupakan fungsi dari kunci dengan panjang yang variabel. Algoritma RC4 memiliki dua fase, yaitu setup kunci dan pengenkripsian. Dengan kunci yang sama maka pada proses dekripsi data kembali ke bentuk semula. Sampai saat ini RC4 masih banyak digunakan orang dalam mengenkripsi informasi berupa data teks karena algoritmanya yang sederhana namun memiliki kecepatan yang hampir sama dibandingkan algoritma yang lebih rumit. Data tersebut berupa .txt yang dienkrip per karakter dengan sebuah kunci yang mengubah plainteks menjadi cipherteks. Dalam aplikasinya datadata penting tidak hanya berupa .txt tetapi ada juga doc, pdf, data gambar dan suara yang sebagian orang merasa perlu mengamankannya sebagai data pribadi. (*Busran ; 2012 : 33*)

RC 4 merupakan jenis stream *cipher*, artinya operasi enkripsi dilakukan per karakter 1 *byte* untuk sekali operasi. Bila dibandingkan dengan block *cipher* yang beroperasi pada data 1 blok (8 *byte* atau 16 *byte*) per operasi enkripsi. Kunci utama RC4 maksimal sepanjang 2048 bit (256 *byte*), namun yang biasa digunakan sepanjang 40 bit atau 128 bit. Sisanya (2048-40=2008 bit atau 2048-128=1920 bit) diisi dengan perulangan kunci tersebut. Jadi jika kuncinya berupa 16 *byte* (128 bit) K= 0123456789abcdef di mana setiap angka merupakan bilangan *hexadesimal* maka byte ke-17 sampai *byte* ke 256 berisi K tersebut secara berulang. RC4 terkenal dengan kecepatan prosesnya. (*Busran ; 2012 : 36*)

Secara garis besar, rangkain proses yang menggambarkan proses enkripsi dan dekripsi data adalah sebagai berikut :



Gambar II.2. Proses Kriptografi
Sumber : (*Busran ; 2012 : 37*)

II.7. *Java*

Java adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer *standalone* ataupun pada lingkungan jaringan *Java* 2 adalah generasi kedua dari *Java platform* (generasi awalnya adalah JDK atau *Java Development Kit*). *Java* inilah yang berdiri diatas mesin *interpreter* yang diberi nama *Java Virtual Machine*(JVM). JVM inilah yang akan membaca *bytecode* dalam *file .class* dari suatu program sebagai representasi langsung program yang berisi bahasa mesin". Oleh karena itu bahasa java disebut juga sebagai bahasa pemrograman yang *portable* karena dapat dijalankan sebagai sistem operasi, asalkan pada sistem operasi tersebut terdapat JVM. (*Utomo Budiyanto ; 2011 : 27*)

Sun Microsystems telah mendefinisikan tiga *platform java* menurut *Utomo Budiyanto 2011* yang masing – masing diarahkan untuk tujuan tertentu dan untuk lingkungan komputasi yang berbeda-beda:

1. *Java Standard Edition* (J2SE), adalah inti dari bahasa pemrograman *java*. JDK adalah salah satu *tool* dari J2SE untuk mengkompilasi program *java* pada JRE.
2. *Java Enterprise Edition* (J2EE), dengan *built-in* mendukung untuk *servlets*, JSP, dan XML, edisi ini ditujukan untuk aplikasi berbasis *server*.
3. *Java Micro Edition* (J2ME), didesain untuk meletakkan perangkat lunak *java* pada barang elektronik beserta perangkat pendukungnya.

Teknologi *Java* mencangkup 2 elemen penting yaitu bahasa pemrograman (*programming language*) dan lingkungan aplikasi (*application environment*). *Java* sebagai bahasa pemrograman dapat diartikan bahwa *java* sebanding dengan bahasa pemrograman seperti C++, *Pascal, Visual Basic*, dan lainya, sedangkan *Java* sebagai lingkungan aplikasi berarti bahwa *java* dapat berjalan pada berbagai lingkungan seperti *browser*(*Applets*), *server*(*servlets* dan JSP) dan pada *mobile device*(*midlet* dan WAP). *Java* dalam hal ini mengungguli bahasa lainnya yang pernah ada jika dilihat dari sisi teknologi *mobile*. Hal ini dibuktikan dengan banyaknya jenis telepon genggam yang menggunakan *java* sebagai fitur utamanya. *Microsoft.NET mobile* pun kelihatannya belum dapat menyaingi keunggulan *Java* dalam bidang aplikasi *mobile*. Perlu diketahui bahwa *Microsoft* hanya mengandalkan solusi WAP yang mengembangkan ASP.NET untuk kebutuhan *mobile device*, sedangkan *Java* memiliki 2 solusi yaitu WAP dan MIDP (*Mobile Information Device Profile*). Solusi pertama adalah dengan mengandalkan J2EE (*Java* 2 *Enterprise Edition*) dengan produknya yang bernama JSP (*Java Server Pages*) dan *Java Servlets*. JSP dan *Servlets* ini digunakan untuk membentuk halaman WAP. Solusi kedua dengan menggunakan J2ME(*Java* 2

Micro Edition) MIDP dengan produknya yang bernama *Midlets*. *Midlets* inilah yang menjadi fitur andalan oleh beberapa jenis telepon genggam terbaru. (*Utomo Budiyanto ; 2011 : 27*)

II.7.1. J2ME (*Java 2 Micro Edition*)

Java Micro Editon atau yang biasa disebut J2ME adalah bagian dari J2SE, karena itu banyak pustaka yang ada pada J2SE dapat digunakan pada J2ME. Tetapi J2ME mempunyai beberapa pustaka khusus yang tidak dimiliki J2SE. Kelahiran *platform* J2ME timbul karena dibutuhkan adanya sebuah *platform* komputasi yang mengakomodasi piranti komputer elektronik dan *embedded*. Piranti ini dikelompokkan menjadi dua kategori, menurut *Utomo Budiyanto ; 2011 : 27* yaitu :

1. Personal, *piranti mobile* yang dapat digunakan untuk komunikasi melalui jaringan tertentu misalkan ponsel, Personal *Digital Assistant* (PDA), *Palm*, *Pocket* PC dan *organizer*.
2. Piranti informasi yang digunakan bersama dengan jaringan tetap, koneksi jaringan yang tidak putus-putus misalnya TV, internet dan sistem navigasi.

Kategori pertama mengarahkan piranti untuk tujuan khusus atau fungsi-fungsi tertentu yang terbatas dan tidak digunakan untuk mesin komputasi yang serba guna. Kategori kedua diarahkan untuk piranti yang mempunyai kapabilitas yang lebih besar dengan fasilitas *user interface* yang lebih baik, kemampuan komputasi yang lebih besar. (*Utomo Budiyanto ; 2011 : 27*)

1. Keunggulan J2ME

Salah satu kelebihan *Java* yang paling signifikan adalah *run everywhere*. Dengan kelebihan ini, para pengembang yang sudah terbiasa mengembangkan aplikasi dalam bingkai kerja J2ME dan J2EE akan mampu bermigrasi dengan mudah untuk mengembangkan aplikasi J2ME. Selain itu, *Java* juga merupakan *platform* yang memiliki banyak keunggulan lain, keunggulan *Java* secara umum adalah :

a. *Multiplatform*, aplikasi J2ME bisa berjalan diatas banyak *platform* yang didalamnya terdapat JVM. Beberapa *platform* yang tersedia didalamnya terdapat JVM antara lain *Windows* CR, *Symbian, Embedded Linux* dan sebagainya.
b. *Robust*, kode-kode *Java* adalah kode-kode *robust*, karena *virtual machine* mengatur keamanan proses eksekusi aplikasi. *Java virtual machine* menyediakan *garbage collector* yang berfungsi mencegah kebocoran memory.
c. Terintegrasi dengan baik, J2ME bisa terhubung dengan *back-end* J2EE *server* dan *web services* dengan mudah karena menyediakan pustaka-pustaka API RMI dan *web services*.
d. Berorientasi obyek, *Java* merupakan salah satu bahasa pemrograman yang murni berorientasi obyek. Hal ini mempermudah dan mempercepat pengembangan sistem yang dikembangkan dengan metode analisa dan desain berorientasi obyek. (*Utomo Budiyanto ; 2011 : 27*)

II.8. UML (*Unified Modelling Language*)

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. 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. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: *Grady Booch OOD* (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*). Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: *metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock*, dsb. Masa itu terkenal dengan

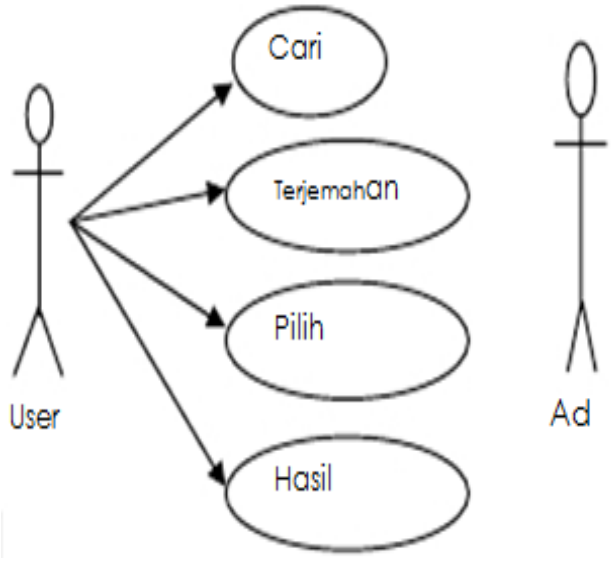
masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan group/perusahaan lain yang menggunakan metodologi yang berlainan. Dimulai pada bulan Oktober 1994 *Booch, Rumbaugh dan Jacobson*, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease *draft* pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh *Object Management Group* (OMG – http://www.omg.org). Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. *Booch, Rumbaugh* dan *Jacobson* menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek. *(Yuni Sugiarti ; 2013 : 33)*

Dalam pembuatan skripsi ini penulis menggunakan diagram *Use Case* yang terdapat di dalam UML. Adapun maksud dari *Use Case* Diagram diterangkan dibawah ini.

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

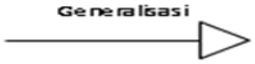
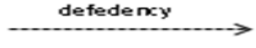
adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* 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. *(Yuni Sugiarti ; 2013 : 41)*



Gambar II.3. ***Use Case Diagram***
Sumber : (*Junaedi Siregar ; 2013 : 76*)

2. *Class Diagram*

Diagram kelas atau *class* diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Berikut adalah simbol-simbol pada diagram kelas :

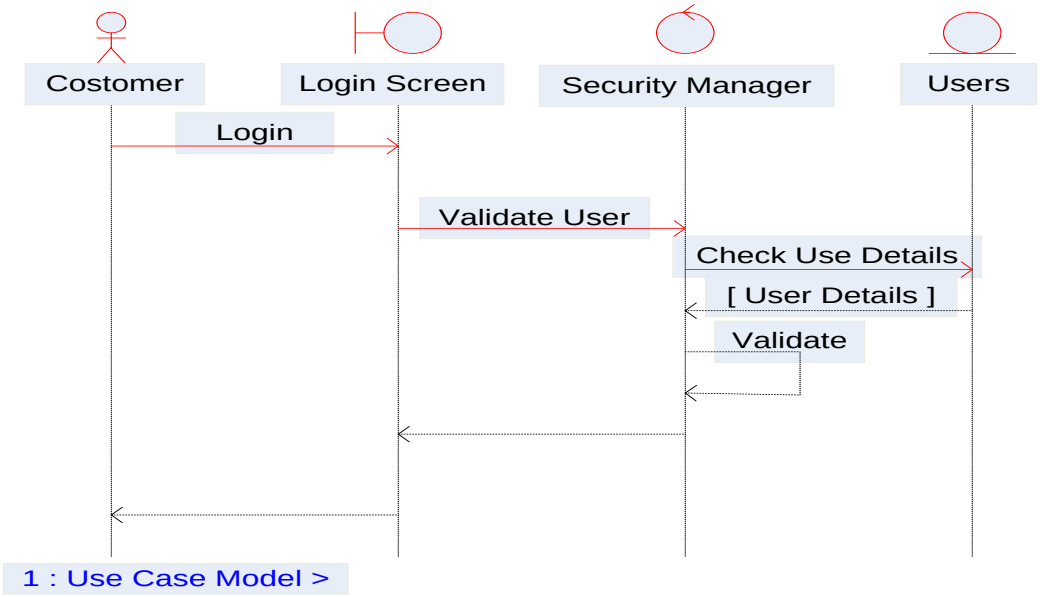
Simbol	Deskripsi
Package 	Package merupakan sebuah bungkusan dari satu atau lebih kelas
Operasi 	Kelas pada struktur sistem
Antarmuka / interface 	sama dengan konsep interface dalam pemrograman berorientasi objek
Asosiasi 	relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan multiplicity
Asosiasi berarah/directed asosiasi	relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan multiplicity
Generalisasi 	relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus)
Kebergantungan / defedency 	relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi 	relasi antar kelas dengan makna semua-bagian (whole-part)

Gambar II.4. *Class* Diagram
Sumber : (Yuni Sugiarti ; 2013 : 59)

3. *Sequence Diagram*

Diagram *Sequence* menggambarkan kelakuan/prilaku objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram *sequence* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya diagram *sequence* yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram *sequence* sehingga semakin banyak *use case* yang didefinisikan maka diagram *sequence* yang harus dibuat juga semakin banyak.



Gambar II.5. *Contoh Sequence Diagram*
Sumber : (Yuni Sugiarti ; 2013 : 63)

4. *Activity Diagram*

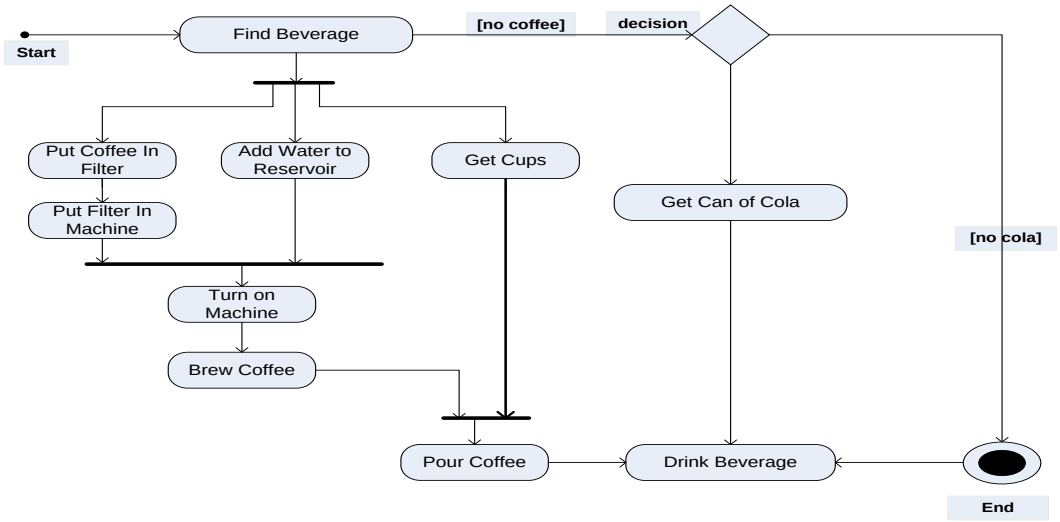
Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

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.

Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal.

Activity diagram dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.



Gambar II.6. *Activify* Diagram
Sumber : (Yuni Sugiarti ; 2013 : 76)