

BAB II

TINJAUAN PUSTAKA

II.1. Aplikasi

Aplikasi merupakan rangkaian kegiatan atau perintah untuk dieksekusi oleh komputer atau suatu perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna. Beberapa aplikasi yang digabung bersama menjadi suatu paket kadang disebut sebagai suatu paket atau suite aplikasi (*application suite*). Aplikasi-aplikasi dalam suatu paket biasanya memiliki antar muka pengguna yang memiliki kesamaan sehingga memudahkan pengguna untuk mempelajari dan menggunakan tiap aplikasi. Sering kali, mereka memiliki kemampuan untuk saling berinteraksi satu sama lain sehingga menguntungkan pengguna. (Ernita Sitohang, 2013).

II.1.1. Data

Data dapat didefinisikan sebagai kenyataan yang digambarkan oleh nilai, bilangan-bilangan, untaian karakter atau simbol-simbol yang membawa arti tertentu. Informasi sendiri dapat didefinisikan sebagai hasil dari pengolahan data dalam bentuk yang lebih berguna bagi penerimanya, yang digunakan sebagai alat bantu dalam pengambilan. (Ernita Sitohang, 2013).

II.1.2. Keamanan

Keamanan merupakan salah satu aspek terpenting dari sebuah sistem informasi. Masalah keamanan sering kurang mendapatkan perhatian dari para perancang dan pengelola sistem informasi. Masalah keamanan sering berada di urutan setelah tampilan, atau bahkan di urutan terakhir dalam daftar hal-hal yang dianggap penting. Keamanan adalah keadaan bebas dari

bahaya. Istilah ini dapat digunakan dengan hubungan kepada kejahatan dan segala bentuk kecelakaan. Keamanan merupakan topik yang luas termasuk keamanan nasional terhadap serangan teroris, keamanan komputer terhadap *hacker*, keamanan rumah terhadap maling dan penyusup lainnya, keamanan financial terhadap kehancuran ekonomi dan banyak situasi berhubungan lainnya. *Host* komputer yang terhubung ke *network*, mempunyai ancaman keamanan lebih besar dari pada *host* yang tidak berhubungan kemana-mana. Dengan mengendalikan *network security* resiko tersebut dapat dikurangi. (Ernita Sitohang, 2013).

II.1.3. Definisi Keamanan Data

Keamanan data dan informasi terdiri dari perlindungan terhadap aspek-aspek berikut:

1. *Confidentiality* (kerahasiaan) aspek yang menjamin kerahasiaan data atau informasi, memastikan bahwa informasi hanya dapat diakses oleh orang yang berwenang dan menjamin kerahasiaan data yang dikirim, diterima dan disimpan.
2. *Integrity* (integritas) aspek yang menjamin bahwa data tidak dirubah tanpa ada ijin pihak yang berwenang (*authorized*), menjaga keakuratan dan keutuhan informasi serta metode prosesnya untuk menjamin aspek *integrity* ini.
3. *Availability* (ketersediaan) aspek yang menjamin bahwa data akan tersedia saat dibutuhkan, memastikan *user* yang berhak dapat menggunakan informasi dan perangkat terkait (aset yang berhubungan bilamana diperlukan).

Keamanan data dan informasi diperoleh dengan mengimplementasikan seperangkat alat kontrol yang layak, yang dapat berupa kebijakan-kebijakan, praktek-praktek, prosedur-prosedur, struktur-stuktur organisasi dan piranti lunak. (Ernita Sitohang, 2013).

II.2. Sistem

Sistem adalah sekumpulan elemen yang saling terkait atau terpadu yang dimaksudkan untuk mencapai suatu tujuan. Sebagai gambaran, jika dalam sebuah sistem terdapat elemen yang tidak memberikan manfaat dalam mencapai tujuan yang sama, maka elemen tersebut dapat dipastikan bukanlah bagian dari sistem. (Abdul Kadir, 2014).

II.2.1. Elemen Sistem

Elemen-elemen yang membentuk sebuah sistem yaitu:

a. Tujuan

Setiap sistem memiliki tujuan (*goal*), entah hanya satu atau mungkin banyak. Tujuan inilah yang menjadi pemotivasi yang mengarahkan sistem. Tanpa tujuan, sistem menjadi tidak terarah dan tidak terkendali. Tentu saja, tujuan antara satu sistem dengan sistem lain berbeda-beda. Begitu pula yang berlaku pada sistem informasi. Setiap sistem informasi memiliki suatu tujuan, tetapi dengan tujuan yang berbeda-beda. Walaupun begitu, tujuan utama yang umum ada tiga macam, yaitu:

1. Untuk mendukung fungsi kepengurusan manajemen,
2. Untuk mendukung pengambilan keputusan manajemen,
3. Untuk mendukung kegiatan operasi perusahaan.

b. Masukan

Masukan (*input*) sistem adalah segala sesuatu yang masuk ke dalam sistem dan selanjutnya menjadi bahan untuk diproses. Masukan dapat berupa hal-hal berwujud (tampak secara fisik) maupun yang tidak tampak. Contoh masukan yang berwujud adalah bahan mentah, sedangkan contoh yang tidak berwujud adalah informasi (misalnya permintaan jasa dari pelanggan). Pada

sistem informasi, masukan dapat berupa data transaksi dan data non-transaksi (misalnya, surat pemberitahuan) serta instruksi.

c. Proses

Proses merupakan bagian yang melakukan perubahan atau transformasi dari masukan menjadi keluaran yang berguna, misalnya berupa informasi dan produk, tetapi juga bisa berupa hal-hal yang tidak berguna, misalnya saja sisa pembuangan atau limbah. Pada pabrik kimia, proses dapat berupa pemanasan bahan mentah. Pada rumah sakit, proses dapat berupa aktivitas pembedahan pasien. Pada sistem informasi, proses dapat berupa suatu tindakan yang bermacam-macam. Meringkas data, melakukan perhitungan dan mengurutkan data merupakan beberapa contoh proses.

d. Keluaran

Keluaran (*output*) merupakan hasil dari pemrosesan. Pada sistem informasi, keluaran bisa berupa suatu informasi, saran, cetakan laporan dan sebagainya. (Abdul Kadir, 2014).

II.3. Kriptografi

Kriptografi pada awalnya dijabarkan sebagai ilmu yang mempelajari bagaimana menyembunyikan pesan. Namun pada pengertian kriptografi modern, kriptografi adalah ilmu yang berdasarkan pada teknik matematika untuk berurusan dengan keamanan informasi seperti kerahasiaan, keutuhan data dan otentikasi entitas. Jadi pengertian kriptografi modern adalah tidak saja berurusan hanya dengan penyembunyian pesan namun lebih pada sekumpulan teknik yang menyediakan keamanan informasi. (Rifki Sadikin, 2012).

1. Fungsi *Hash*

Fungsi *hash* adalah fungsi yang melakukan pemetaan pesan dengan panjang sembarang ke sebuah teks khusus yang disebut *message digest* dengan panjang tetap. Fungsi *hash* umumnya dipakai sebagai nilai uji (*check value*) pada mekanisme keutuhan data. (Rifki Sadikin, 2012)

2. Penyandian Dengan Kunci Simetrik

Penyandian dengan kunci simetrik adalah penyandian yang kunci enkripsi dan kunci dekripsi bernilai sama. Kunci pada penyandian simetrik diasumsikan bersifat rahasia hanya pihak yang melakukan enkripsi dan dekripsi yang mengetahui nilainya. Oleh karena itu penyandian dengan kunci simetrik disebut juga penyandian dengan kunci rahasia *secret key encipherment*. (Rifki Sadikin, 2012)

3. Penyandian dengan kunci asimetrik

Penyandian dengan kunci asimetrik atau sering disebut juga dengan penyandian kunci publik (*public key*) adalah penyandian dengan kunci enkripsi dan dekripsi berbeda nilai. Kunci enkripsi yang juga disebut dengan kunci publik (*public key*) bersifat terbuka. Sedangkan kunci dekripsi yang juga disebut kunci privat (*private key*) bersifat tertutup / rahasia. (Rifki Sadikin, 2012).

II.3.1. Sistem Kriptografi

Sistem kriptografi terdiri dari 5 bagian yaitu:

1. *Plaintext*

Pesan atau data dalam bentuk aslinya yang dapat terbaca. *Plaintext* adalah masukan bagi algoritma enkripsi. (Rifki Sadikin, 2012).

2. *Secret Key*

Secret key yang juga merupakan masukan bagi algoritma enkripsi merupakan nilai yang bebas terhadap teks asli dan menentukan hasil keluaran algoritma enkripsi. (Rifki Sadikin, 2012).

3. *Ciphertext*

Ciphertext adalah keluaran algoritma enkripsi. *Ciphertext* dapat dianggap sebagai pesan dalam bentuk tersembunyi. Algoritma enkripsi yang baik akan menghasilkan *ciphertext* yang terlihat acak. (Rifki Sadikin, 2012).

4. Algoritma Enkripsi

Algoritma enkripsi memiliki 2 masukan teks asli dan kunci rahasia. Algoritma enkripsi melakukan transformasi terhadap teks asli sehingga menghasilkan teks sandi. (Rifki Sadikin, 2012).

5. Algoritma Dekripsi

Algoritma dekripsi memiliki 2 masukan yaitu teks sandi dan kunci rahasia. Algoritma dekripsi memulihkan kembali teks sandi menjadi teks asli bila kunci rahasia yang dipakai algoritma dekripsi sama dengan kunci rahasia yang dipakai algoritma enkripsi. (Rifki Sadikin, 2012).

II.3.2. Karakteristik Sistem Kriptografi

Sistem kriptografi dapat dikarakteristikan berdasarkan:

1. Tipe Operasi Dipakai Dalam Enkripsi Dan Dekripsi

Dua tipe operasi yang dipakai dalam enkripsi dan dekripsi: substitusi, elemen pada pesan (karakter, *byte* atau *bit*) ditukar / disubstitusikan dengan elemen lain dari ruang pesan. Misalnya substitusi sederhana A ditukar B, B ditukar D, dan C ditukar Z, pesan "BACA" menjadi "DBZB". Tipe operasi lainnya adalah transposisi, elemen pada pesan berpindah posisi misalnya posisi 1 menjadi posisi 4 dan posisi 2 menjadi posisi 3, posisi 3 menjadi posisi 1 dan posisi 4 menjadi

posisi 2, pesan "KAMI" menjadi "MAIK". Sistem kriptografi modern mencakup kedua tipe operasi ini. (Rifki Sadikin, 2012).

2. Tipe Kunci Yang Dipakai

Umumnya sistem kriptografi klasik dan beberapa sistem kriptografi modern menggunakan kunci yang sama pada sisi penyandi dan penyulih sandi. Sistem kriptografi seperti ini disebut kriptografi dengan kunci simetri. Baru pada tahun 1976 sistem kriptografi yang memperbolehkan kunci yang tidak sama diusulkan oleh Whitfield Diffie dan Martin Hellman, (Diffie & Hellman, 1976). Sistem kriptografi ini disebut dengan kriptografi dengan kunci asimetri. (Rifki Sadikin, 2012).

3. Tipe Pengolahan Pesan

Ketika melakukan penyandian pesan yang akan dienkripsi ataupun didekripsi diolah pesatuan blok elemen disebut dengan *block cipher*. Cara lain adalah dengan menganggap masukan untuk enkripsi dan dekripsi sebagai aliran elemen secara terus menerus disebut dengan *stream cipher*. (Rifki Sadikin, 2012).

II.4. SMS (*Short Message Service*)

SMS (*Short Message Service*) merupakan sebuah layanan komunikasi yang ada pada telepon seluler untuk mengirim dan menerima pesan-pesan pendek. SMS pertama kali dikenalkan pada tanggal 3 Desember 1982. SMS pertama di dunia dikirimkan menggunakan jaringan GSM milik operator telepon bernama *Vodafone*. SMS pertama ini dikirimkan oleh ahli bernama Neil Papworth kepada Richard Jarvis menggunakan komputer. (Andi Kurniawan Dwi P, 2012).

II.5. OTP (*One Time Pad*)

Algoritma *vernam cipher* atau biasa dikenal dengan sebutan *one time pad (OTP)* merupakan algoritma berjenis *symmetric key* yang artinya bahwa kunci yang digunakan untuk melakukan enkripsi dan dekripsi merupakan kunci yang sama. Dalam proses enkripsi, algoritma ini menggunakan cara *stream cipher* yang berasal dari hasil *XOR* antara bit *plaintext* dan *bit key*. Pada metode ini *plain text* diubah kedalam kode *biner* dan kemudian dikenakan operasi *XOR* terhadap kunci yang sudah diubah ke dalam kode *biner*. (M. Sholeh dan J.V. Hamokwarong, 2011).

II.6. Android

Android adalah sistem operasi untuk perangkat selular yang berbasis *Linux*. *Android* menyediakan platform terbuka bagi para pengembang buat menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak. Awalnya, *Google Inc* membeli *Android Inc* pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan *Android*, dibentuklah *Open Handset Alliance*, konsorsium dari 34 perusahaan peranti keras, peranti lunak dan telekomunikasi, termasuk *Google*, *HTC*, *Intel*, *Motorola*, *Qualcomm*, *T-Mobile*, dan *Nvidia*. Pada saat perilisan perdana *Android*, 5 November 2007, *Android* bersama *Open Handset Alliance* menyatakan mendukung pengembangan standar terbuka pada perangkat seluler. Di lain pihak, *Google* merilis kode-kode *Android* di bawah lisensi *Apache*, sebuah lisensi perangkat lunak dan standar terbuka perangkat seluler. Di dunia ini terdapat dua jenis distributor sistem operasi. Salah satu *software* pengembang sistem *Android* adalah *eclipse*. Pada penelitian ini peneliti menggunakan *eclipse galileo*. (Andi Sanjaya, 2014).

Pada saat perilisan perdana *Android*, 5 November 2007, *Android* bersama *Open Handset Alliance* menyatakan mendukung pengembangan *open source* pada perangkat *mobile*. Di lain

pihak, *Google* merilis kode-kode *Android* dibawah lisensi *Apache*, sebuah lisensi perangkat lunak dan *open platform* perangkat seluler. (Nazruddin Safaat H, 2014).

II.6.1. Versi *Android*

Telepon pertama yang memakai sistem operasi *Android* adalah *HTC Dream*, yang dirilis pada 22 oktober 2008. Pada penghujung tahun 2010 diperkirakan hampir semua vendor seluler didunia menggunakan *Android* sebagai *operating system*. Adapun versi-versi *Android* yang pernah dirilis adalah sebagai berikut:

1. *Android* Versi 1.1

Pada 9 Maret 2009, *Google* merilis *Android* versi 1.1. *Android* versi ini dilengkapi dengan pembaruan estetis pada aplikasi, jam, alarm, *voice search* (pencarian suara), pengiriman pesan dengan *Gmail* dan pemberitahuan *email*.



Gambar II.1. *Android* Versi 1.1
(Sumber: https://en.wikipedia.org/wiki/Android_version_history)

2. *Android* Versi 1.5 (*Cupcake*)

Pada pertengahan Mei 2009, *Google* kembali merilis telepon seluler dengan menggunakan *Android* dan *SDK* (*Software Development Kit*) dengan versi 1.5 (*Cupcake*). Terdapat beberapa pembaruan termasuk juga penambahan beberapa fitur dalam seluler versi ini yakni kemampuan merekam dan menonton *video* dengan modus kamera, mengupload *video* ke *youtube* dan gambar ke *picasa* langsung dari telepon, dukungan *Bluetooth A2DP*, kemampuan terhubung secara otomatis ke *headset Bluetooth*, animasi layar dan *keyboard* pada layar yang dapat disesuaikan dengan sistem.



Gambar II.2. Android Versi 1.5 (Cupcake)
(Sumber: https://en.wikipedia.org/wiki/Android_version_history)

3. *Android Versi 1.6 (Donut)*

Donut (versi 1.6) dirilis pada September dengan menampilkan proses pencarian yang lebih baik dibanding sebelumnya, penggunaan baterai indikator dan kontrol *applet VPN*. Fitur lainnya adalah galeri yang memungkinkan pengguna untuk memilih foto yang akan dihapus, kamera, *camcorder* dan galeri yang diintegrasikan, *CDMA/EVDO*, 802.1x, *VPN*, *Gestures*, dan *Text-to-speech engine*, kemampuan *dial* kontak, teknologi *text to change speech* (tidak tersedia pada semua ponsel, pengadaan resolusi *VWGA*).



Gambar II.3. Android Versi 1.6 (Donut)
(Sumber: https://en.wikipedia.org/wiki/Android_version_history)

4. *Android Versi 2.0 / 2.1 (Eclair)*

Pada 3 Desember 2009 kembali diluncurkan ponsel *Android* dengan versi 2.0 / 2.1 (*Éclair*), perubahan yang dilakukan adalah pengoptimalan *hardware*, peningkatan *Google Maps* 3.1.2, perubahan *UI* dengan *browser* baru dan dukungan *HTML5*, daftar kontak yang baru, dukungan *flash* untuk kamera 3,2 *MP* dan *digital zoom*, *bluetooth* 2.1.



Gambar II.4. Android Versi 2.0 / 2.1 (Enclair)
(Sumber: https://en.wikipedia.org/wiki/Android_version_history)

5. *Android Versi 2.2 (Froyo: Frozen Yoghurt)*

Pada bulan mei 2010 *Android* vers 2.2 Rev 1 diluncurkan. *Android* inilah yang sekarang banyak beredar di pasaran, salah satunya adalah dipakai di *Samsung FX tab* yang sudah ada di pasaran. Fitur yang tersedia di *Android* versi ini sudah kompleks diantaranya adalah:

- a. Kerangka aplikasi memungkinkan penggunaan dan penghapusan komponen yang tersedia.
- b. *Dalvik Virtual Machine* dioptimalkan untuk perangkat *mobile*.
- c. Grafik: grafik di 2D dan grafis 3D berdasarkan *libraries OpenGL*.
- d. *SQLite*: untuk penyimpanan data.
- e. Mendukung media: *audio*, *video* dan berbagai *format* gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).
- f. *GSM*, *Bluetooth*, *EDGE*, *3G* dan *Wifi* (*hardware independent*).
- g. Kamera, *Global Positioning System (GPS)*, kompas dan *accelerometer* (tergantung *hardware*).



Gambar II.5. Android Versi 2.2 (Froyo: Frozen Yoghurt)
(Sumber: https://en.wikipedia.org/wiki/Android_version_history)

6. *Android Versi 2.3 (Gingerbread)*

Android versi 2.3 diluncurkan pada desember 2010, hal-hal yang direvisi dari versi sebelumnya adalah kemampuan seperti berikut:

- a. *SIP-based VoIP*

- b. *Near Field Communications (NFC)*
- c. *Gyroscope dan sensor*
- d. *Multiple cameras support*
- e. *Mixable audio effects*
- f. *Download Manager*



Gambar II.6. Android Versi 2.3 (Gingerbread)
(Sumber: https://en.wikipedia.org/wiki/Android_version_history)

7. *Android Versi 3.0 (Honeycomb)*

Dirilis Februari 2011 sebagai *Android 3.0* revisi 1 serta *Android* versi 3.0 *revision 2* telah dirilis pada juli 2011.



Gambar II.7. Android Versi 3.0 (Honeycomb)
(Sumber: https://en.wikipedia.org/wiki/Android_version_history)

8. *Android Versi 3.1*

Dirilis Mei 2011, sedangkan *Android 3.1* revisi 2 juga dirilis mei 2011, serta *Android 3.1* revision 3 dirilis pada juli 2011.



Gambar II.8. Android Versi 3.1
(Sumber: https://en.wikipedia.org/wiki/Android_version_history)

9. *Android* Versi 3.2

Dirilis Juli 2011.



Gambar II.9. Android Versi 3.2
(Sumber: https://en.wikipedia.org/wiki/Android_version_history)

10. *Android* Versi 4.0

Dirilis November 2011.



Gambar II.10. Android Versi 4.0
(Sumber: https://en.wikipedia.org/wiki/Android_version_history)

11. *Android* Versi 4.1



Gambar II.11. Android Versi 4.1
(Sumber: https://en.wikipedia.org/wiki/Android_version_history)

12. *Android* Versi 4.2



Gambar II.12. Android Versi 4.2

(Sumber: https://en.wikipedia.org/wiki/Android_version_history)

13. *Android* Versi 4.3

Android versi 3.0 ke atas adalah generasi *platform* yang digunakan untuk *tablet pc*. Sementara versi 4.0 sudah merupakan *platform* yang bisa dipakai di *smartphone* dan *tablet pc*. (Nazruddin Safaat H, 2014).



Gambar II.13. *Android* Versi 4.3

(Sumber: https://en.wikipedia.org/wiki/Android_version_history)

II.6.2. *The Dalvik Virtual Machine (DVM)*

Salah satu elemen kunci dari *Android* adalah *Dalvik Virtual Machine (DVM)*, *Android* berjalan di dalam *Dalvik Virtual Machine (DVM)* bukan di *Java Virtual Machine (JVM)*, sebenarnya banyak persamaannya dengan *Java Virtual Machine (JVM)* seperti *Java ME (Java Mobile Edition)*, tetapi *Android* menggunakan *Virtual Machine* sendiri yang menurut saya dikustomisasi dan dirancang untuk memastikan bahwa beberapa *feature-feature* berjalan lebih efisien pada perangkat *mobile*. (Nazruddin Safaat H, 2014).

Dalvik Virtual Machine (DVM) adalah "*register bases*" sementara *Java Virtual machine (JVM)* adalah "*stack based*", *DVM* di *design* dan ditulis oleh dan bornsten dan beberapa *engineers google* lainnya. jadi bisa kita katakan "*Dalvik equals (java) = false*". *Dalvik virtual machine* menggunakan kernel linux untuk menangani fungsional tingkat rendah termasuk keamanan, *threading*, serta proses *managemen* memori. Hal ini memungkinkan kita menulis aplikasi C / C++ sama halnya seperti pada *OS linux* kebanyakan. Meskipun dalam kenyataannya

kita harus banyak memahami arsitektur dan proses sistem dari kernel *linux* yang digunakan dalam *android* tersebut. (Nazruddin Safaat H, 2014).

Semua *hardware* yang berbasis *android* dijalankan dengan menggunakan *Virtual Machine* untuk eksekusi aplikasi, sehingga pengembang tidak perlu khawatir tentang implementasi perangkat keras tertentu. *Dalvik Virtual machine* mengaksekusi *executable file*, sebuah format yang dioptimalkan untuk memastikan memori yang digunakan sangat kecil. *The executable file* diciptakan dengan mengubah kelas bahasa *java* dan kompilasi menggunakan *tools* yang disediakan dalam *SDK Android* (Nazruddin Safaat H, 2014).

II.6.3. Android SDK (Software Development Kit)

Android SDK adalah *tools API (Application Programming Interface)* yang diperlukan untuk mulai mengembangkan aplikasi pada *platform Android* menggunakan bahasa pemrograman *Java*. *Android* merupakan *subset* perangkat lunak untuk ponsel yang meliputi sistem operasi, *middleware* dan aplikasi kunci yang di *release* oleh *Google*. Saat ini disediakan *Android SDK (Software Development Kit)* sebagai alat bantu dan *API* untuk mulai mengembangkan aplikasi pada *platform Android* menggunakan bahasa pemrograman *Java*. Sebagai *platform* aplikasi-netral, *Android* member ada kesempatan untuk membuat aplikasi yang kita butuhkan yang bukan merupakan aplikasi bawaan *Handphone / Smartphone*. (Nazruddin Safaat H, 2014).

II.6.4. ADT (Android Development Tools)

Android Development Tools (ADT) adalah *plugin* yang didesain untuk *IDE Eclipse* yang memberikan kita kemudahan dalam mengembangkan aplikasi *Android* dengan menggunakan *IDE Eclipse*. Dengan menggunakan *ADT* untuk *Eclipse* akan memudahkan kita dalam membuat aplikasi *project Android*, membuat *GUI* aplikasi dan menambahkan komponen-komponen yang

lainnya, begitu juga kita dapat melakukan *running* aplikasi menggunakan *Android SDK* melalui *eclipse*. Dengan *ADT* kita juga dapat melakukan pembuatan *package Android* (.apk) yang digunakan untuk distribusi aplikasi *Android* yang kita rancang. (Nazruddin Safaat H, 2014).

II.7. Sistem

Sistem adalah sekumpulan elemen yang saling terkait atau terpadu yang dimaksudkan untuk mencapai suatu tujuan. Sebagai gambaran, jika dalam sebuah sistem terdapat elemen yang tidak memberikan manfaat dalam mencapai tujuan yang sama, maka elemen tersebut dapat dipastikan bukanlah bagian dari sistem. (Abdul Kadir, 2014).

II.7.1. Elemen Sistem

Elemen-elemen yang membentuk sebuah sistem yaitu:

a. Tujuan

Setiap sistem memiliki tujuan (*goal*), entah hanya satu atau mungkin banyak. Tujuan inilah yang menjadi pemotivasi yang mengarahkan sistem. Tanpa tujuan, sistem menjadi tidak terarah dan tidak terkendali. Tentu saja, tujuan antara satu sistem dengan sistem lain berbeda-beda. Begitu pula yang berlaku pada sistem informasi. Setiap sistem informasi memiliki suatu tujuan, tetapi dengan tujuan yang berbeda-beda. Walaupun begitu, tujuan utama yang umum ada tiga macam, yaitu:

1. Untuk mendukung fungsi kepengurusan manajemen,
2. Untuk mendukung pengambilan keputusan manajemen,
3. Untuk mendukung kegiatan operasi perusahaan.

b. Masukan

Masukan (*input*) sistem adalah segala sesuatu yang masuk ke dalam sistem dan selanjutnya menjadi bahan untuk diproses. Masukan dapat berupa hal-hal berwujud (tampak secara fisik)

maupun yang tidak tampak. Contoh masukan yang berwujud adalah bahan mentah, sedangkan contoh yang tidak berwujud adalah informasi (misalnya permintaan jasa dari pelanggan). Pada sistem informasi, masukan dapat berupa data transaksi, dan data non-transaksi (misalnya, surat pemberitahuan) serta instruksi.

c. Proses

Proses merupakan bagian yang melakukan perubahan atau transformasi dari masukan menjadi keluaran yang berguna, misalnya berupa informasi dan produk, tetapi juga bisa berupa hal-hal yang tidak berguna, misalnya saja sisa pembuangan atau limbah. Pada pabrik kimia, proses dapat berupa pemanasan bahan mentah. Pada rumah sakit, proses dapat berupa aktivitas pembedahan pasien. Pada sistem informasi, proses dapat berupa suatu tindakan yang bermacam-macam. Meringkas data, melakukan perhitungan dan mengurutkan data merupakan beberapa contoh proses.

d. Keluaran

Keluaran (*output*) merupakan hasil dari pemrosesan. Pada sistem informasi, keluaran bisa berupa suatu informasi, saran, cetakan laporan dan sebagainya. (Abdul Kadir, 2014).

II.7.2. Informasi

McFadden, dkk. (1999) Mendefinisikan informasi sebagai data yang telah diproses sedemikian rupa sehingga meningkatkan pengetahuan seseorang yang menggunakan data tersebut. Shannon dan Weaver, dua orang insinyur listrik, melakukan pendekatan secara matematis untuk mendefinisikan informasi (Kroenke, 1992). Menurut mereka, informasi adalah “jumlah ketidakpastian yang dikurangi ketika sebuah pesan diterima”. Artinya, dengan adanya sistem informasi, tingkat kepastian menjadi meningkat. Menurut Davis (1999), informasi adalah

data yang telah diolah menjadi sebuah bentuk yang berarti bagi penerimanya dan bermanfaat dalam pengambilan keputusan saat ini atau saat mendatang. (Abdul Kadir, 2014).

II.8. Java

Java adalah sebuah bahasa pemrograman yang populer dikalangan para akademisi dan praktisi komputer. *Java* pertama kali dikembangkan untuk memenuhi kebutuhan akan sebuah bahasa komputer yang ditulis satu kali dan dapat dijalankan dibanyak sistem komputer berbeda tanpa perubahan kode berarti. Pada umumnya, para pakar pemrograman berpendapat bahwa bahasa *Java* memiliki konsep yang konsisten dengan teori pemrograman objek dan aman untuk digunakan.

Java sampai saat ini masih merupakan bahasa pemrograman yang masih sangat diminati dan banyak digunakan oleh para *programer* dan *software developer* untuk mengembangkan berbagai tipe aplikasi, mulai dari aplikasi *console*, aplikasi *desktop*, *game* dan *applet* (aplikasi yang berjalan di lingkungan *web browser*), sampai ke aplikasi-aplikasi yang berskala *enterprise*. Untuk memenuhi kebutuhan tipe aplikasi yang beragam tersebut, *Java* dikategorikan menjadi tiga edisi, yaitu: *J2SE (Java 2 Platform Standart Edition)* untuk membuat aplikasi-aplikasi *desktop* dan *applet*, *J2EE (Java 2 Platform Enterprise Edition)* untuk membuat aplikasi-aplikasi *multitier* berskala *enterprise* dan *J2ME (Java 2 Platform Micro Edition)* untuk membuat aplikasi-aplikasi yang dapat dijalankan dilingkungan perangkat-perangkat mikro seperti *handphone*, *PDA* dan *Smartphone*. (Retno Wardhani dan Moh Husnul Yaqin, 2013).

II.9. Unified Modeling Language (UML)

Menurut Windu Gata (2013) Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *Unified Modeling Language (UML)*. *UML* adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak.

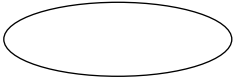
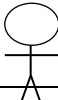
UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. *UML* saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem. (Gellysa Urva dan Helmi Fauzi Siregar, 2015).

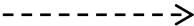
Alat bantu yang digunakan dalam perancangan berorientasi objek berbasis *UML* adalah sebagai berikut:

1. *Use case* Diagram

Use case diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram dapat dilihat pada tabel II.1. berikut:

Tabel II.1. Simbol *Use Case*

Gambar	Keterangan
	<i>Use case</i> menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor dan dinyatakan dengan menggunakan kata kerja di awal nama <i>use case</i> .
	Aktor adalah <i>abstraction</i> dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasi aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa

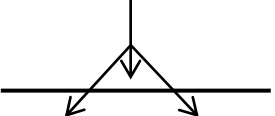
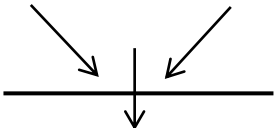
	muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan <i>use case</i> , tetapi tidak memiliki kontrol terhadap <i>use case</i> .
	Asosiasi antara aktor dan <i>use case</i> , digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasikan aliran data.
	Asosiasi antara aktor dan <i>use case</i> yang menggunakan panah terbuka untuk mengindikasikan bila aktor berinteraksi secara pasif dengan sistem.
	<i>Include</i> , merupakan di dalam <i>use case</i> lain (<i>required</i>) atau pemanggilan <i>use case</i> oleh <i>use case</i> lain, contohnya adalah pemanggilan sebuah fungsi program.
	<i>Extend</i> , merupakan perluasan dari <i>use case</i> lain jika kondisi atau syarat terpenuhi.

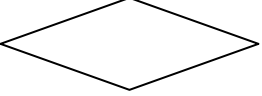
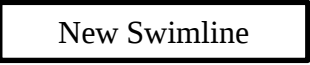
(Sumber: Gellysa Urva dan Helmi Fauzi Siregar; 2015)

2. Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram* dapat dilihat pada tabel II.2. berikut:

Tabel II.2. Simbol *Activity Diagram*

Gambar	Keterangan
	<i>Start point</i> , diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	<i>End point</i> , akhir aktifitas.
	<i>Activites</i> , menggambarkan suatu proses/kegiatan bisnis.
	<i>Fork</i> (percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan parallel menjadi satu.
	<i>Join</i> (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.

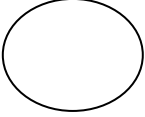
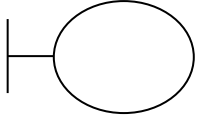
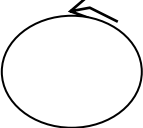
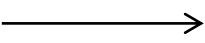
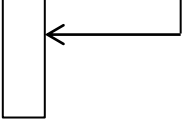
	<i>Decision Points</i> , menggambarkan pilihan untuk pengambilan keputusan, <i>true</i> , <i>false</i> .
	<i>Swimlane</i> , untuk menunjukkan siapa melakukan apa.

(Sumber: Gellysa Urva dan Helmi Fauzi Siregar; 2015)

3. Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram* dapat dilihat pada tabel II.3. berikut:

Tabel II.3. Simbol *Sequence Diagram*

Gambar	Keterangan
	<i>Entity Class</i> , merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	<i>Boundary Class</i> , berisi kumpulan kelas yang menjadi <i>interface</i> atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan <i>formentry</i> dan <i>form</i> cetak.
	<i>Control class</i> , suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
	<i>Message</i> , simbol mengirim pesan antar <i>class</i> .
	<i>Recursive</i> , menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	<i>Activation</i> , <i>activation</i> mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	<i>Lifeline</i> , garis titik-titik yang terhubung dengan objek, sepanjang <i>lifeline</i> terdapat <i>activation</i> .

(Sumber: Gellysa Urva dan Helmi Fauzi Siregar; 2015)

4. *Class Diagram* (Diagram Kelas)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. *Class diagram* juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*), *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau *kardinaliti* yang dapat dilihat pada tabel II.4. berikut:

Tabel II.4. *Multiplicity Class Diagram*

Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

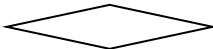
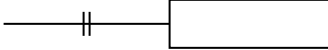
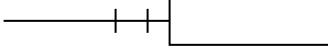
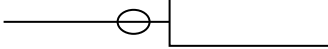
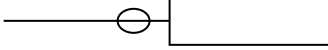
(Sumber: Gellysa Urva dan Helmi Fauzi Siregar; 2015)

II.10. *Entity Relationship Diagram (ERD)*

Entity Relationship Diagram (ERD) adalah bagian yang menunjukkan hubungan antara *entity* yang ada dalam sistem. Simbol-simbol yang digunakan dapat dilihat dari tabel II.5. (Yuhendra, M.T, Dr. Eng dan Riza Eko Yulianto, 2015).

Tabel II.5. Simbol Yang Digunakan Pada *Entity Relationship Diagram (ERD)*

SIMBOL	KETERANGAN
---------------	-------------------

	<i>Entity</i>
	Atribut Dan <i>Entity</i>
	Atribut Dan <i>Entity</i> Dengan <i>Key</i> (Kunci)
	Relasi Atau Aktivitas Antar <i>Entity</i>
	Hubungan Satu Dan Pasti
	Hubungan Banyak Dan Pasti
	Hubungan Satu Tapi Tidak Pasti
	Hubungan Banyak Tapi Tidak Pasti

(Sumber: Yuhendra, M.T, Dr. Eng dan Riza Eko Yulianto; 2015)