BAB II

TINJAUAN PUSTAKA

II.1. Pengertian Implementasi Sistem Keamanan

Implementasi adalah suatu prosedur yang dilakukan untuk menyelesaikan sistem yang ada dalam dokumen rancangan sistem yang telah disetujui dan telah diuji, menginstal dan memulai menggunakan sistem baru yang diperbaiki. Tahap ini merupakan tahap inti dari pekerjaaan sebuah proyek. Implementasi sistem yang dimaksud merupakan proses pembuatan dan pemasangan sistem secara utuh baik dari sisi *hardware*, *software* dan *brainware*-nya (Zaid Amin, 2012).

Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran tertentu (Zaid Amin, 2012).

Keamanan adalah suatu kinerja dalam menghadapi masalah baik internal maupun eksternal yang terjadi terhadap suatu ruang lingkup demi terciptanya suatu keadaan yang seharusnya (Sentot Kromodimoeljo, 2010).

Dari pengertian di atas dapat disimpulkan bahwa implementasi sistem keamanan merupakan suatu proses pembuatan dan pemasangan jaringan kerja dari prosedur-prosedur yang saling berhubungan untuk menyelesaikan suatu kinerja dalam menghadapi masalah baik internal maupun eksternal yang terjadi terhadap suatu ruang lingkup demi terciptanya suatu keadaan yang seharusnya.

Aspek-aspek yang berkaitan dengan persyaratan keamanan antara lain:

1. *Secrecy*. Berhubungan dengan akses membaca data dan informasi. Data dan informasi di dalam suatu sistem komputer hanya dapat diakses dan dibaca oleh orang yang berhak.
2. *Integrity*. Berhubungan dengan akses merubah data dan informasi. Data dan informasi yang berada di dalam suatu sistem komputer hanya dapat diubah oleh orang yang berhak.
3. *Availability*. Berhubungan dengan ketersediaan data dan informasi. Data dan informasi yang berada dalam suatu sistem komputer tersedia dan dapat dimanfaatkan oleh orang yang berhak.

(Sentot Kromodimoeljo, 2010).

II.2. Pengertian Kriptografi

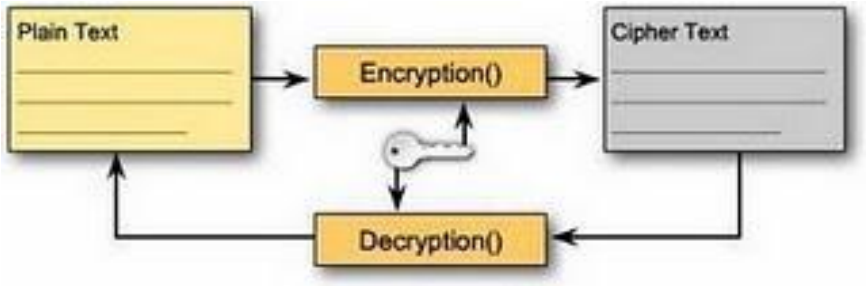
Kriptografi (*cryptography*) merupakan ilmu dan seni penyimpanan pesan, data, atau informasi secara aman. Kriptografi (*cryptography*) berasal dari bahasa Yunani yaitu dari kata *cryptós* yang artinya *secret* (yang tersembunyi) dan *gráphia* yang artinya *writting* (tulisan). Menurut terminologinya kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan ketika pesan dikirim dari suatu tempat ketempat lain (Sentot Kromodimoeljo, 2010).

Jadi, kriptografi berarti *secret writting* (tulisan rahasia). Kriptografi merupakan bagian dari suatu cabang ilmu matematika yang disebut *Cryptology* yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data. Kriptografi bertujuan menjaga kerahasiaan informasi yang terkandung dalam data sehingga informasi

tersebut tidak dapat diketahui oleh pihak yang tidak sah. Tetapi tidak semua aspek keamanan informasi ditangani oleh kriptografi (Sentot Kromodimoeljo, 2010).

II.2.1. Algoritma Kriptografi

Secara sederhana proses kriptografi dapat digambarkan sebagai berikut:



Gambar II.1. Ilustrasi dari proses kriptografi sederhana
(Sumber: *Teori dan Aplikasi Kriptografi*, Sentot Kromodimoeljo, 2010)

Dalam menjaga kerahasiaan data, kriptografi mentransformasikan data asli (*plaintext*) ke dalam bentuk data sandi (*ciphertext*) yang tidak dapat dikenali. *Ciphertext* inilah yang kemudian dikirimkan oleh pengirim (*sender*) kepada penerima (*receiver*). Setelah sampai di penerima, *ciphertext* tersebut di tranformasikan kembali ke dalam bentuk *plaintext* agar dapat dikenali. Proses tranformasi dari plaintext menjadi ciphertext disebut proses *Encipherment* atau enkripsi (*encryption*), sedangkan proses mentransformasikan kembali *ciphertext* menjadi *plaintext* disebut proses dekripsi (*decryption*). Kunci (*key*) adalah parameter yang digunakan untuk transformasi *enciphering* dan *deciphering* (Sentot Kromodimoeljo, 2010).

Secara umum operasi enkripsi dan dekripsi dapat diterangkan secara matematis sebagai berikut:

Proses enkripsi ditulis dalam Persamaan 1:

E (P) = C ..... (1)

Dimana,

E : Enkripsi

P : Pesan (*plaintext*) yang akan diamankan

C : *Chipertext*

Sedangkan proses dekripsi ditulis dalam Persamaan 2 :

D (C) = P ..... (2)

Dimana,

D : Dekripsi

C : *Chipertext* yang akan dirubah menjadi plaintext

P : Pesan (*plaintext*) yang telah dibuka

Karena proses enkripsi dan dekripsi mengembalikan pesan ke pesan awal, maka Persamaan 3 harus benar,

D (E(P)) = P ..... (3)

Pada saat proses enkripsi kita menyandikan pesan P dengan suatu kunci K lalu dihasilkan pesan C. Sedangkan pada proses dekripsi, pesan C tersebut diuraikan dengan menggunakan kunci K sehingga dihasilkan pesan P yang sama seperti pesan sebelumnya (Sentot Kromodimoeljo, 2010).

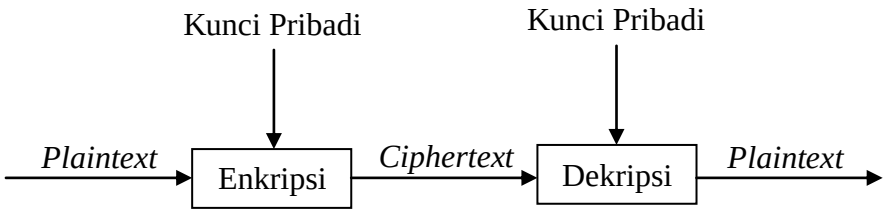
Dengan demikian keamanan suatu pesan tergantung pada kunci ataupun kunci-kunci yang digunakan dan tidak tergantung pada algoritma yang digunakan. Sehingga algoritma-algoritma yang digunakan tersebut dapat dipublikasikan dan dianalisis, serta produk-produk yang menggunakan algoritma tersebut dapat

diproduksi masa. Tidaklah menjadi masalah apabila seseorang mengetahui algoritma yang digunakan. Selama tidak mengetahui kunci yang digunakan maka tetap tidak dapat membaca pesan (Sentot Kromodimoeljo, 2010).

Berdasarkan kunci yang dipakai, algoritma kriptografi dapat dibedakan atas dua golongan yaitu kriptografi kunci simetris dan kriptografi kunci asimetris (Sentot Kromodimoeljo, 2010).

II.2.2 Kriptografi Kunci Simetris

Algoritma kriptografi simetris atau disebut juga algoritma kriptografi konvensioanl adalah algoritma yang menggunakan kunci untuk proses enkripsi sama dengan kunci untuk proses dekripsi, oleh karena itulah dinamakan kriptografi simetris. Keamanan sistem kriptografi simetris terletak pada kerahasiaan kuncinya. Ada banyak algoritma kriptografi modern yang termasuk ke dalam sistem kriptografi simetris, di antaranya adalah DES (*Data Encryption Standard*), *Blowfish*, *Twofish*, *Triple*-DES, IDEA, Serpent, AES (*Advanced Encryption Standard*) (Sentot Kromodimoeljo, 2010).



Gambar II.2. Diagram proses enkripsi dan dekripsi algoritma simetris (Sumber: Sentot Kromodimoeljo, 2010)

Sebelum melakukan pengiriman pesan, pengirim dan penerima harus memilih suatu kunci tertentu yang sama untuk dipakai bersama, dan kunci ini

haruslah rahasia bagi pihak yang tidak berkepentingan sehingga algoritma ini disebut juga algoritma kunci rahasia (*secret-key algorithm*) (Sentot Kromodimoeljo, 2010).

Kelebihan dari algoritma kunci simetris atau disebut juga dengan algoritma kunci rahasia adalah:

1. Kecepatan operasi lebih tinggi bila dibandingkan dengan algoritma asimetris.
2. Karena kecepatannya yang cukup tinggi, maka dapat digunakan pada sistem *real-time*.

Sedangkan kelemahan dari algoritma kunci simetris ini yaitu:

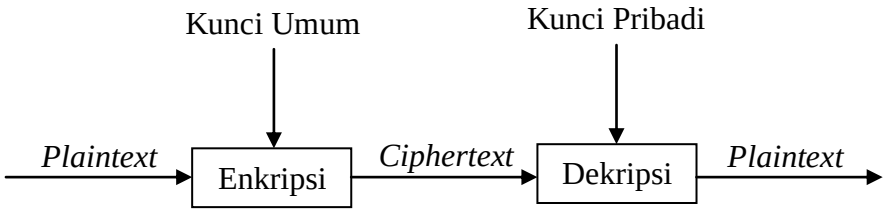
1. Untuk tipe pengiriman pesan dengan pengguna yang berbeda dibutuhkan kunci yang berbeda juga, sehingga akan terjadi kesulitan dalam manajemen kunci tersebut.
2. Permasalahan dalam pengiriman kunci itu sendiri yang disebut "*key distribution problem*".

Contoh algoritma: Blowfish, Rijndael, Camellia, DES, AES (Sentot Kromodimoeljo, 2010).

II.2.3. Kriptografi Kunci Asimetris

Algoritma kriptografi asimetris adalah algoritma yang menggunakan kunci yang berbeda untuk proses enkripsi dan dekripsinya. Algoritma ini disebut juga algoritma kunci umum (*public key algorithm*) karena kunci untuk enkripsi dibuat umum (*public key*) atau dapat diketahui oleh setiap orang, tapi kunci untuk dekripsi hanya diketahui oleh orang yang berwenang mengetahui data yang

disandikan atau sering disebut kunci pribadi (*private key*). Contoh algoritma terkenal yang menggunakan kunci asimetris adalah RSA dan ECC (Sentot Kromodimoeljo, 2010).



Gambar II.3. Diagram proses enkripsi dan dekripsi algoritma asimetris (Sumber: Sentot Kromodimoeljo, 2010)

Dalam *Asymmetric Algorithm* ini digunakan dua buah kunci. Satu kunci yang disebut kunci publik (*public key*) dapat dipublikasikan, sedangkan kunci yang lain yang disebut kunci privat (*private key*) harus dirahasiakan. Proses penggunaan sistem ini dapat diterangkan secara sederhana sebagai berikut: bila A ingin mengirimkan pesan kepada B, A dapat menyandikan pesannya dengan menggunakan kunci publik B, dan bila B ingin membaca surat tersebut, ia perlu mendekripsikan surat itu dengan kunci privatnya. Dengan demikian kedua belah pihak dapat menjamin asal surat serta keaslian surat tersebut karena adanya mekanisme ini. Contoh algoritma asimetris antara lain *RSA Scheme* dan *Merkle-Hellman Scheme* (Sentot Kromodimoeljo, 2010).

II.3. Algoritma One Time Pad

One Time Pad adalah salah satu contoh metode kriptografi dengan algoritma jenis simetri. Sehingga kunci yang digunakan untuk proses enkripsi sama dengan kunci yang digunakan untuk proses dekripsi. Ditemukan pada tahun

1917 oleh Major Yoseph Mouborgne dan Gilbert Vernam pada perang dunia ke dua. Metode ini telah diklaim sebagai satu-satunya algoritma kriptografi sempurna yang tidak dapat dipecahkan. Suatu algoritma dikatakan aman, apabila tidak ada cara untuk menemukan plaintext-nya Sampai saat ini, hanya algoritma One Time Pad (OTP) yang dinyatakan tidak dapat dipecahkan meskipun diberikan sumber daya yang tidak terbatas. Proses enkripsi dan dekripsi pada One Time Pad ini hampir sama dengan proses enkripsi dan dekripsi menggunakan algoritma vigenere cipher (Sugeng Sutrisno, 2014). Proses enkripsi dapat dilakukan dengan persamaan matematis sebagai berikut :

$$C_i = (P_i + K_i) \; Mod \; 26$$

Sedangkan untuk proses dekripsi dapat dilihat pada persamaan matematis sebagai berikut:

$$P_i = ((C_i - K_r) + 26) \; Mod \; 26$$

Dari persamaan di atas dapat diketahui :

C_i = pergeseran karakter pada ciphertext

P_i = pergeseran karakter padaplaintext

K_r = Kunci dalam bentuk decimal yang dihasilkan dari tabel konversi.

Bagian yang membedakan antara one time pad dengan vigenere cipher adalah pada kunci yang digunakan. Jika penggunaan kunci pada vigenere cipher dapat diulang untuk menyesuaikan dengan panjang plaintext, maka pada one time pad hal tersebut tidak dapat dilakukan karena jumlah kunci yang digunakan harus sama panjangnya dengan jumlah plaintext (Sugeng Sutrisno, 2014).

Sebagai contoh penyelesaian algoritma one time pad adalah sebagai berikut:

Misalkan pesan yang akan dikirimkan yaitu FIRMAN dengan kata kunci GLORIA, Langkah yang dilakukan sbb :

1. Ubah menjadi kode ASCII dan biner untuk kata FIRMAN dan GLORIA, sehingga didapat hasil sebagai berikut:

Karakter	ASCII	BINER
F	106	0001 0000 0110
I	111	0001 0001 0001
R	122	0001 0010 0010
M	115	0001 0001 0101
A	101	0001 0000 0001
N	116	0001 0001 0110

Hal yang sama dilakukan pada kunci

G	107	0001 0000 0111
L	114	0001 0001 0100
O	117	0001 0001 0111
R	122	0001 0010 0010
I	111	0001 0001 0001
A	101	0001 0000 0001

2. Pesan di-XORkan dengan kunci maka akan diperoleh

F 0000 0000 0001001

I 0000 0000 0101005

R 0000 0011 0101035

M 0000 0011 0100034

A 0000 0001 0000010

N 0000 0001 0111017

3. Kode ASCII tersebut diterjemahkan lagi menjadi karakter Diperoleh : NUL ENQ GS FS BS SI
4. Untuk memperoleh plainteks kembali, penerima pesan cukup mengubah lagi plainteks menjadi ASCII dan meng-XORkan kembali dengan kunci.

II.4 Email

Electronic Mail atau biasa yang disebut Email merupakan salah satu jenis service yang tidak bisa dilepaskan dari segala aktivitas yang terjadi di dunia maya, penggunaan email bukan hanya untuk bisnis semata tapi juga untuk social network, pengiriman pesan secara cepat, pengiriman file yang penting dan hal lainnya yang berkaitan dengan proses transfer informasi. *Microsoft Outlook* dan *Mozilla Thunderbird* merupakan aplikasi yang di khususkan untuk pengiriman dan penerimaan email dari pihak ketiga, kemudahan pengiriman dan penerimaan email sehingga bisa dibaca offline tanpa harus terhubung ke internet (Reza Fahlevi, 2014).

II.5 Unified Modelling Language

Menurut Yuni Sugiarti (2013:34) *Unified Modelling Language* (UML) adalah sebuah bahasa yang menjadi standar dalam industri untuk visualisasi,

merancang dan mendokumentasikan sistem piranti lunak. UML biasa digunakan untuk :

1. Menggambarkan batasan sistem dan fungsi-fungsi sistem secara umum, dibuat dengan *user* dan *actor*
2. Menggambarkan kegiatan atau proses bisnis yang dilaksanakan secara umum, dibuat dengan *interactions diagrams*
3. Menggambarkan respresentasi struktur statik sebuah sistem dalam bentuk *class diagram*
4. Membuat model behavior ”yang menggambarkan kebiasaan atau sifat sebuah sistem” dengan *state transition diagrams.*

UML merupakan salah satu alat bantu yang sangat handal dalam bidang pengembangan system berorientasi objek karena *UML* menyediakan bahasa pemodelan visual yang memungkinkan pengembang sistem membuat *blue print* atas visinya dalam bentuk yang baku (Yuni Sugiarti, 2013:34).

UML berfungsi sebagai jembatan dalam mengkomunikasikan beberapa aspek dalam sistem melalui sejumlah elemen grafis yang bias dikombinasikan menjadi diagram. *UML* mempunyai banyak diagram yang dapat mengkomodasi berbagai sudut pandang dari suatu perangkat lunak yang akan dibangun (Yuni Sugiarti, 2013:34).

II.6. Diagram Dasar dalam UML

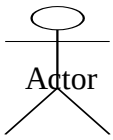
II.6.1. *Use Case Diagram*

Use Case Diagram atau diagram use case merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem yang akan dibuat. Diagram use case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Dengan pengertian yang cepat, diagram use case digunakan untuk mengetahui fungsi apa saja yang ada dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Terdapat beberapa simbol dalam menggambarkan diagram use case, yaitu use case, aktor dan relasi. Hal yang perlu diingat mengenai diagram use case adalah diagram use bukan menggambarkan tampilan antar muka (*user interface*), arsitektur dari sistem, kebutuhan nonfungsional, dan tujuan performansi. Sedangkan untuk penamaan *use case* adalah nama didefinisikan sesimple mungkin, dapat dipahami dan menggunakan kata kerja (Yuni Sugiarti : 2013 : 41).

II.6.1.1 Simbol-Simbol pada *Use Case Diagram*

Berikut adalah simbol-simbol yang ada pada diagram use case :

Tabel II.1 : Simbol-simbol pada *use case diagrams*

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
 Aktor	Orang, proses, atau sistem lain yang berinteraksi sebagai 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
Asosiasi / Association	Komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor
<<Extend>> Extend	Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan itu, miri dengan prinsip *inheritance* pada pemrograman berorientasi objek, biasanya use case tambahan memiliki nama depan yang sama dengan use case yang ditambahkan, arah panah menunjuk use case yang dituju.
<<Include>> Include	Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan memerlukan 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, include berarti use case yang ditambahkan akan selalu dipanggil saat use case tambahan dijalankan

Sumber : Yuni Sugiarti (2013 : 42)

II.6.2. *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. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas, atribut mendeskripsikan properti dengan sebaris teks didalam kotak kelas tersebut. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Diagram kelas mendeskripsikan jenis-jenis objek dalam sistem dan berbagai hubungan statis yang terdapat diantara mereka. Diagram juga

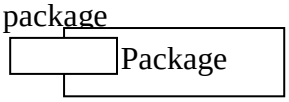
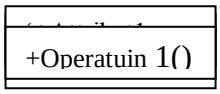
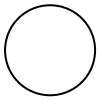
menunjukkan properti dan operasi sebuah kelas dan batasan-batasan yang terdapat dalam hubungan-hubungan objek tersebut.

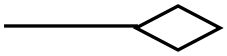
Diagram kelas menggambarkan struktur dan deskripsi kelas, package, dan objek berserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain-lain. Kelas memiliki tiga area pokok : nama, atribut dan operasi (Yuni Sugiarti : 2013 : 57).

II.6.2.1. Simbol-Simbol pada *Class Diagram*

Berikut adalah simbol-simbol yang ada pada diagram kelas :

Tabel II.2 : Simbol-simbol *class diagram*

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 / Association 1 1..*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
Aosiasi berarah/directed asosiasi	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas lain, asosiasi biasanya juga disertau 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)

Sumber : Yuni Sugiarti (2013 : 59)

II.6.3. *Sequence Diagram*

Diagram Sequence menggambarkan kelakuan/perilaku 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 case beserta metode-metode yang dimiliki kelas yang di instansiasi menjadi objek itu.

Diagram sequence memiliki ciri yang berbeda dengan diagram interaksi pada diagram kolaborasi sebagai berikut :

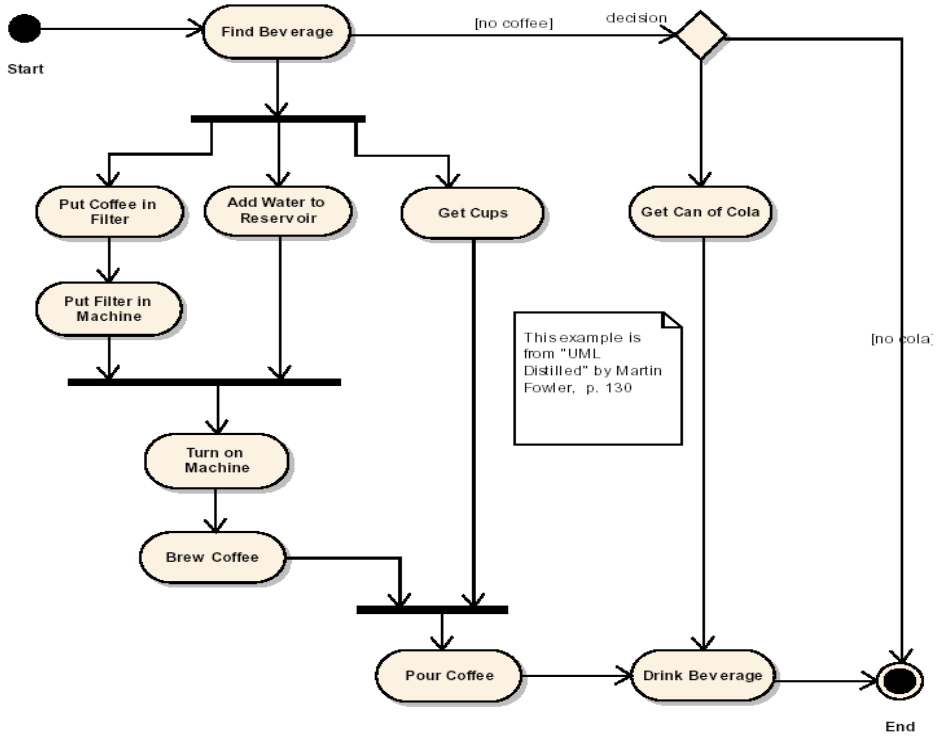
1. Pada diagram sequence terdapat garis objek
2. Terdapat pokus kendali (*Focus of Control)* , berupa empat persegi panjang ramping dan tinggi yang menampilkan aksi atau objek secara langsung atau sepanjang sub ordinat (Yuni Sugiarti : 2013 : 69-70).

II.6.4. *Activity Diagram*

Digram aktivitas atau *Activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa

yang dilakukan aktor, jadi aktivitas yang dapat dilakukakn oleh sistem. Diagram aktivitas mendukung perilaku paralel.

Activity diagrams 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 (Yuni Sugiarti : 2013 : 75).



Gambar. II.4. Activity Diagram Tanpa Swimlalane
(Sumber: Yuni Sugiarti (2013 : 76)

II.7. *Microsoft Visual Basic Net 2010 (VB Net 2010*)

Microsoft Visual Basic .NET adalah sebuah alat untuk mengembangkan dan membangun aplikasi yang bergerak di atas sistem .NET Framework, dengan

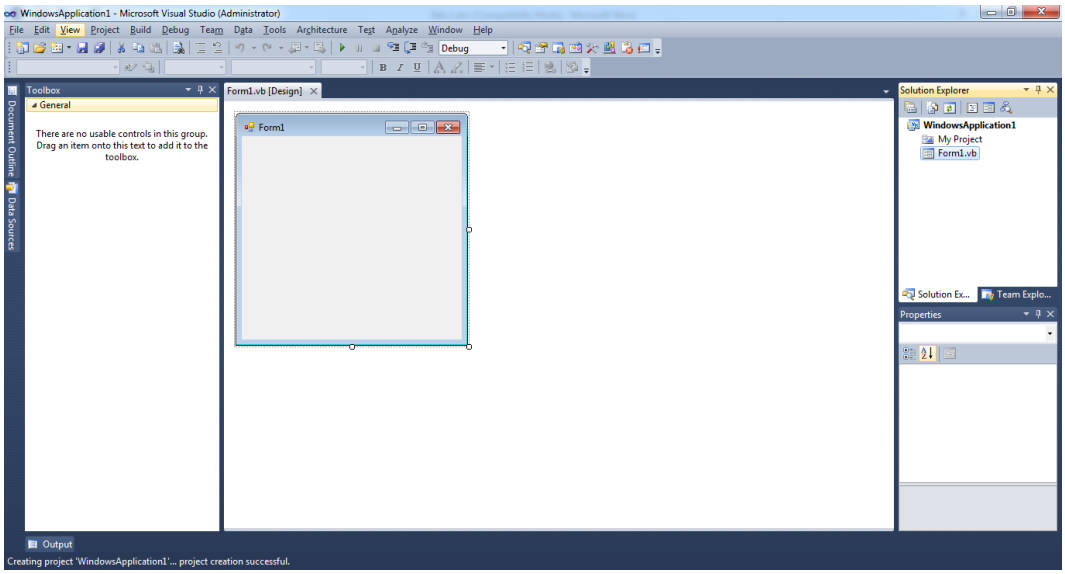
menggunakan bahasa BASIC. Dengan menggunakan alat ini, para programmer dapat membangun aplikasi Windows Forms, Aplikasi web berbasis ASP.NET, dan juga aplikasi command-line. Alat ini dapat diperoleh secara terpisah dari beberapa produk lainnya (seperti Microsoft Visual C++, Visual C#, atau Visual J#), atau juga dapat diperoleh secara terpadu dalam Microsoft Visual Studio .NET. Bahasa Visual Basic .NET sendiri menganut paradigma bahasa pemrograman berorientasi objek yang dapat dilihat sebagai evolusi dari Microsoft Visual Basic versi sebelumnya yang diimplementasikan di atas .NET Framework. Peluncurannya mengundang kontroversi, mengingat banyak sekali perubahan yang dilakukan oleh Microsoft, dan versi baru ini tidak kompatibel dengan versi terdahulu (Wahana Komputer, 2012).

Apakah Visual Basic .NET dianggap sebagai sebuah versi Visual Basic atau benar-benar bahasa yang berbeda merupakan sebuah topik perdebatan yang hangat. Hal ini dikarenakan sintaksis bahasa Visual Basic .NET tidak mengalami perubahan yang sangat drastis, dan hanya menambahkan beberapa dukungan fitur baru seperti penanganan eksepsi secara terstruktur dan ekspresi yang bisa di-short-circuit-kan. Dua perubahan tipe data pun terjadi saat berpindah ke Visual Basic .NET. Dibandingkan dengan Visual Basic 6.0, tipe data Integer yang dimiliki oleh Visual Basic .NET memiliki panjang dua kali lebih panjang, dari 16 bit menjadi 32 bit. Selain itu, tipe data Long juga sama-sama berubah menjadi dua kali lipat lebih panjang, dari 32 bit menjadi 64 bit. Bilangan bulat 16-bit dalam Visual Basic .NET dinamakan dengan Short. Lagi pula, desainer GUI Windows Forms yang terdapat di dalam Visual Studio .NET atau Visual Basic .NET memiliki gaya

yang sangat mirip dengan editor form Visual Basic klasik (Wahana Komputer, 2012).

Jika sintaksis tidak banyak yang berubah, lain halnya dengan semantik, yang berubah secara signifikan. Visual Basic .NET merupakan sebuah bahasa pemrograman yang mendukung fitur "Bahasa Pemrograman Berorientasi Objek" secara penuh, karena memang didukung oleh arsitektur Microsoft .NET Framework, yang mengandung kombinasi dari Common Language Runtime dan Base Class Library. Visual Basic klasik, hanya merupakan sebuah bahasa pemrogaman berbasis objek, yang berjalan di atas arsitektur Component Object Model (COM) (Wahana Komputer, 2012).

Perubahan ini telah mengubah banyak asumsi tentang hal yang benar yang harus dilakukan dengan mempertimbangkan performa dan kemudahan untuk dipelihara. Beberapa fungsi dan pustaka perangkat lunak, yang ada di dalam Visual Basic klasik, kini tidak terdapat di dalam Visual Basic .NET; mungkin masih banyak yang masih terdapat di dalam Visual Basic .NET, tapi tidak seefisien apa yang ditawarkan oleh .NET Framework. Bahkan jika program Visual Basic klasik bisa dikompilasi dengan benar, sebagian besar program Visual Basic klasik harus melalui beberapa proses refactoring untuk mengadopsi fitur bahasa baru secara keseluruhan. Dokumentasi untuk ini pun tersedia di situs Microsoft (Wahana Komputer, 2012).



Gambar II.5. Visual Basic.Net

Sumber : Wahana Komputer, 2012