BAB II

TINJAUAN PUSTAKA

II.1. Aplikasi

Aplikasi berasal dari kata *application* yang artinya penerapan, lamaran, penggunaan. Secara istilah aplikasi adalah program siap pakai yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain dan dapat digunakan oleh sasaran yang dituju. Perangkat lunak aplikasi adalah suatu subkelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan tugas yang diinginkan pengguna. Contoh utama perangkat lunak aplikasi adalah pengolah kata, lembar kerja, dan pemutar media (Fricles Ariwisanto Sianturi; 2013: 43).

II.2. Kriptografi

Kriptografi (*cryptography*) berasal dari bahasa yunani, yaitu dari kata *crypto* dan *graphia* yang berarti penulisan rahasia. Kriptografi adalah ilmu ataupun seni yang mempelajari bagaimana membuat suatu pesan yang dikirim oleh pengirim dapat disampaikan kepada penerima dengan aman. Kriptografi merupakan bagian dari suatu cabang ilmu matematika yang disebut kriptologi *(cryptology).* Kriptografi bertujuan menjaga kerahasiaan informasi yang terkandung dalam data sehingga informasi tersebut tidak dapat diketahui oleh pihak yang tidak sah. Perancang algoritma kriptografi disebut *kriptografer*. (Emy Setyaningsih; 2015: 2).

Kriptografi (*cryptography*) berasal dari Bahasa Yunani: *"cryptós"* artinya *"secret"* (rahasia), sedangkan *"gráphein"* artinya *"writing"* (tulisan). Jadi, kriptografi berarti *"secret writing"* (tulisan rahasia). Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data, serta otentikasi (Mariana; 2011: 2).

Kriptografi merupakan seni dan ilmu menyembunyikan informasi dari penerima yang tidak berhak. Kata kriptografi berasal dari kata Yunani *kryptos* (tersembunyi) dan *graphein* (menulis). Dalam teknologi informasi telah dan sedang dikembangkan cara-cara untuk menangkal berbagai serangan, seperti penyadap dan pengubahan data yang sedang dikirimkan. Transformasi ini memberikan solusi pada dua macam masalah keamanan data, yaitu masalah privasi *(privacy*) dan keotentikan (*authenticatioan*). Kriptografi tidak berarti hanya memberikan keamanan informasi saja, namun lebih ke arah teknik-tekniknya.

a. Enkripsi

Proses enkripsi adalah proses penyandian pesan terbuka (*plaintext*) menjadi pesan rahasia (*ciphertext*). *Ciphertex*t inilah yang nantinya akan dikirimkan melalui saluran komunikasi terbuka. Pada saat *ciphertext* diterima oleh penerima pesan, maka pesan rahasia tersebut diubah lagi menjadi pesan terbuka melalui proses dekripsi sehingga pesan tadi dapat dibaca kembali oleh penerima pesan.

b. Dekripsi

Dekripsi merupakan proses kebalikan dari proses *enkripsi*, merubah *ciphertext* kembali ke dalam bentuk *plaintext*. Untuk menghilangkan penyandian yang diberikan pada saat proses

enkripsi, membutuhkan penggunaan sejumlah informasi rahasia, yang disebut sebagai kunci (Fricles Ariwisanto Sianturi; 2013: 43).

II.3 Database

Basis data (*Database*) adalah suatu pengorganisasian sekumpulan data yang saling terkait sehingga memudahkan aktivitas untuk memperoleh informasi. Basis data dimaksudkan untuk mengatasi problem pada sistem yang memakai pendekatan berbasis berkas (Abdul Kadir, 2010:254).

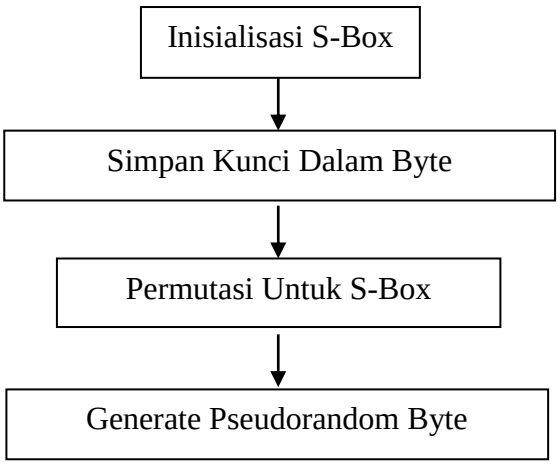
Menurut Eko Koswara (2011), *Database adalah* kumpulan data-data dalam bentuk tabel yang saling berhubungan yang disimpan dalam media perangkat keras (contohnya :Harddisk) yang dapat diambil lagi sebagai informasi.Elemen-elemen penyusun database antara lain adalah:

1. Tabel, merupakan kumpulan record dengan fomat field yang sama, satu tabel biasanya mempresentasikan data satu objek maupun satu kejadian yang terjadi dalam sebuah sistem. Contoh tabel pegawai yang menyimpan data mengenai objek pegawai, tabel transaksi pembelian yang menyimpan data-data mengenai kejadian transaksi pembelian.
2. Field/Kolom, merupakan bagian terkecil dari tabel yang digunakan untuk menyimpan informasi item. Contoh *field ID* digunakan untuk menyimpan *item* informasi berupa nomor induk pegawai. *Field* Nama digunakan untuk menyimpan item tanggal transaksi yang dilakukan pada suatu transaksi.
3. *Record/Baris*, merupakan sekumpulan *field* yang berhubungan erat, menggambarkan suatu informasi.

4. *Primary Key*, merupakan suatu field yang nilainya unik dan digunakan sebagai kunci yang membedakan record satu dengan *record* lainnya. Contoh tabel Mahasiswa, primary keynya adalah *field* NIM (Nomor Induk Mahasiswa)
5. *Relationship*, hubungan antar satu tabel dengan tabel lainnya.
6. *Query*, digunakan untuk menyaring dan menampilkan data yang memenuhi kriteria tertentu dalam satu tabel atau lebih. Query dilakukan dengan menggunakan bahasa *SQL* (*Strukture Query language*)

II.4. Algoritma RC4

RC4 adalah cipher aliran yang digunakan secara luas pada sistem keamanan seperti *protocol secure socket layer* (SSL). Algoritma kriptografi ini sederhana dan mudah diimplementasikan. RC4 dibuat oleh Ron Rivest di Laboratorium *RSA* (*RC* adalah singkatan dari *Ron's Code*). Gambar 1.1 memperlihatkan rangkaian proses yang dijalankan untuk mengenkripsi atau mendeskripsi data.



Gambar I1.1 Rangkaian Proses RC4

(Sumber : Emy Setyaningsih 126)

RC4 membangkitkan keystream yang kemudian di XOR kan dengan plaintext pada waktu enkripsi (atau di XOR kan dengan bit bit ciphertext pada waktu dekripsi). RC4 tidak seperti cipher aliran yang memproses data dalam bit. RC4 memproses data dalam ukuran byte (satu byte = 8 bit). (Emy Setyaningsih; 2015: 126).

RC4 merupakan salah satu jenis cipher aliran (*stream cipher*) , didesain oleh Ron Rivest di Laboratorium RSA (*RSA Data Security Inc.*) pada tahun 1987. Cipher RC4 merupakan teknik enkripsi yang dapat dijalankan dengan panjang kunci yang variabel dan memproses data dalam ukuran *byte*. .Kriptografi modern beroperasi pada mode bit, yang berarti semua data dan informasi (kunci, *plainteks*, maupun *cipherteks*) dinyatakan dalam rangkaian (*string*) *bit biner*, 0 dan 1. (Basuki Rakhmat;2012;4).

II.5. Vigenere

Vigenere atau vigenere cipher adalah metode enkripsi abjad majemuk manual (*polyalphabetical substitution cipher*). Algoritma ini ditemukan oleh seorang diplomat sekaligus *kriptolog prancis, blaise de vigenere*, pada abad XVI. Metode ini dipublikasikan pada tahun 1856, dan sekitar 200 tahun setelahnya, pada abad XIX, *vigenere* cipher digunakan oleh tentara konfederasi pada perang sipil amerika.

Vigenere cipher pada dasarnya menggunakan teknik yang sama dengan Caesar cipher. Bedanya, dalam *vigenere cipher* setiap karakter pada *plaintext* dapat dienkripsikan dengan kunci

yang berbeda. Karakter pertama pada *plaintext* dienkripsikan dengan kunci berupa karakter pertama dari kata kunci dan seterusnya. Sifat *polialfabetik* yang dimiliki oleh *vigenere* ciper di implementasikan dengan bujursangkar vigenere. Sifat periodiknya terlihat apabila panjang kunci lebih kecil daripada panjang plaintext. Kunci dapat diulang penggunaanya sampai panjang kunci sama dengan panjang plaintext. Jika panjang kunci hanya satu karakter, enkripsinya sama dengan caisar cipher biasa. (Emy Setyaningsih; 2015: 85).

Vigenere atau *Vigenere cipher* merupakan *cipher* yang setiap *plaintext*-nya mempunyai beberapa kemungkinan *ciphertext*, ini terjadi karena panjang kuncinya lebih dari satu. *Cipher* ini mempunyai fungsi matematika yang sama dengan *caesar cipher*, yaitu :

C = (P + K) mod n P = (C - K) mod n C = *ciphertext* K = kunci P = *plaintext* n = jumlah karakter

Jika hasil dekripsi (C - K) bernilai negatif (-), maka nilai ditambahkan dengan jumlah karakter (n). Bagaimana membangun suatu aplikasi kriptografi yang kuat dengan mengkombinasikan metode *stream cipher* dan *vigenere cipher* untuk melakukan enkripsi dan dekripsi. (Wiwiek Nurwiyati:2013;153).

II.6. Visual Basic .Net

Visual basic 2010 merupakan salah satu bagian dari produk pemrograman terbaru yang dikeluarkan oleh *Microsoft*, yaitu *microsoft Visual Studio* 2010. Sebagai produk lingkungan pengembangan terintegrasi atau IDE andalan yang di keluarkan oleh *microsoft, visual studio* 2010 menambahkan perbaikan-perbaikan fitur dan fitur baru yang lebih lengkap *visual studio* pendahuluanya, yaitu *mirosoft visual studio* 2008. (Wahana Komputer;2010;2)

Sedangkan menurut Aswan (2012 : 1) *Visual basic* 2010 adalah salah satu bagian dari *microsoft visual studio* 2010. Sebuat alat yang digunakan oleh pengembang windows dari berbagai level untuk mengembangkan dan membangun aplikasi yang bergerak diatas sistem .NET Framework, dengan menggunakan bahasa BASIC. Visual Basic menyediakan cara cepat dan mudah untuk membuat aplikasi.

Setiap generasi baru dari perangkat lunak bahasa pemrograman datang karena adanya keterbatasan dari generasi sebelumnya. Teknologi *device, hardware, network* dan internet baru yang muncul menyebabkan bahasa pemrograman yang ada tidak lagi menjadi alat yang ideal untuk mengembangkan perangkat lunak yang dapat bekerja dengan teknologi baru tersebut (WAH[12]). Sekarang untuk pertama kalinya, platform pengembang perangkat lunak yang lengkap, *Microsoft .NET* telah didesain dari dasar dengan internet sebagai fokus utamanya (walaupun tidak secara eksklusif hanya untuk pengembang internet saja). Banyak inovasi baru yang berada dalam platform ini akan mengatasi keterbatasan dari tool-tool dan teknologi lama. *Visual Basic .NET* adalah pengembangan dari *Visual basic* sebelumnya. Kelebihan *VB .NET* 2010 terletak pada tampilannya yang lebih canggih dibandingkan dengan edisi Visual Basic sebelumnya. Selain memiliki kelebihan, *VB .NET* 2005 memiliki kekurangan. Kekurangan *VB .NET* 2005 yang terlihat jelas adalah beratnya aplikasi ini apabila dijalankan pada komputer yang memiliki spesifikasi sederhana.

II.7. UML *(Unified Modeling Language)*

UML(*Unified Modeling Language)* yang merupakan metodologi kolaborasi antara metoda booch, OMT (*Object Modeling Technique),* serta OOSE (*Oriented Sofware Engineering)* dan beberapa metoda lainnya, merupakan metodologi yang paling sering digunakan saat ini

untuk mengadaptasi maraknya penggunaan bahasa “pemrograman berorientasi objek” (OOP). (Adi Nugroho;2009;4)

UML (*Unified Modeling Language*) adalah sebuah ”bahasa” yang telah menjadi standar dalam industry untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Seperti bahasa-bahasa lainnya, UML mendefenisikan notasi dan *sintax/semantic*. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *sintax* mendefinisikan bagaimana bentuk – bentuk tersebut dapat dikombinasikan. *Unified Modeling Language* biasa digunakan untuk :

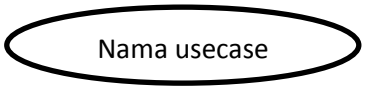
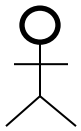
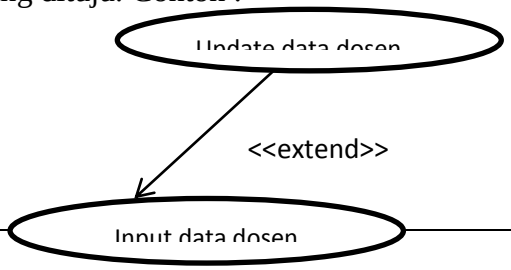
1. Menggambarkan batasan sistem dan fungsi – fungsi sistem secara umum, di buat dengan *use case* dan *actor*.
2. Menggambarkan kegiatan atau proses bisnis yang di laksanakan secara umum, di buat dengan *interaction diagrams*.
3. Menggambarkan representasi struktur *static* sebuah sistem dalam bentuk *class diagram*s.
4. Membuat model behavior “yang menggambarkan kebiasaan atau sifat sebuah sistem” dengan *state transition diagrams*.
5. Menyatakan arsitektur implementasi fisik menggunakan *component and development diagrams*.
6. Menyampaikan atau memperluas *fungsionality* dengan *stereotypes*. (Yuni Sugiarti; 2013 :36)

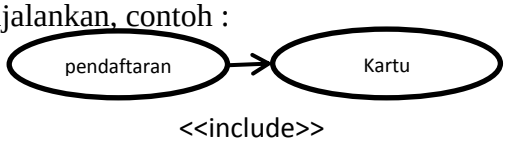
II.7.1. Use Case Diagram

Use case diagrams merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem yang akan dibuat. Diagram *use case* mendeskripsikan sebuah interaksi antara satu atau

lebih *actor* dengan sistem yang akan dibuat. Dengan pengertian yang cepat, diagram *use case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi – fungsi tersebut. Terdapat beberapa simbol dalam menggambarkan diagram *use case*, yaitu *use case*, *actor* dan relasi. Berikut adalah sismbol – simbol yang ada pada diagram *use case*. (Yuni Sugiarti; 2013: 42)

Tabel II.1 Simbol – simbol pada *Use Case Diagram*

Simbol	Deskripsi
Use case 	Fungsionalitas yang disediakan sistem sebagai unit – unit yang saling bertukar pesan antar unit atau *actor*; biasanya ditanyakan dengan menggunakan kata kerja di awal frase nama *use case*.
Aktor  nama aktor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari *acto*r adalah gambar orang, tapi *actor* belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda diawal frase nama *actor*.
Asosiasi/ *association* 	Komunikasi antara actor daan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan kator.
Extend  <<extend>>	Relasi use case tambahan ke sebuah 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, arah panah menunjuk pada use case yang dituju. Contoh : 

Include <<include>> ⟶	Relasi use case tambahan kesebuah use case dimana use case yang 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, contoh :  <<include>>

Sumber: (Yuni Sugiarti; 2013)

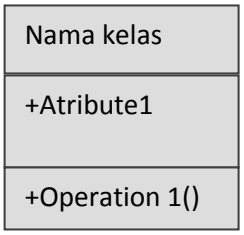
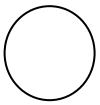
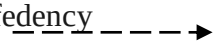
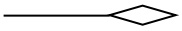
II.7.2. *Class* Diagram

Diagram kelas atau class diagram menggambarkan struktur sistem dari segi pendefenisian kelas – kelas yang akan di buat untuk membangun sistem. Kelas memiliki apa yang di sebut atribut dan metode atau operasi.

1. Atribut merupakan variabel- variabel yang di miliki oleh suatu kelas.
2. Atribut mendeskripsikan properti dengan sebaris teks di dalam kotak kelas tersebut.
3. Operasi atau metode adalah fungsi – fungsi yang di miliki oleh suatu kelas.

Diagram kelas mendeskripsikan jenis – jenis objek dalam sistem dan berbagai hubungan statis yang terdapat di antara mereka. Diagram kelas juga menunjukkan properti dan operasi sebuah kelas dan batasan – batasan yang terdapat dalam hubungan – hubungan objek tersebut. (Yuni Sugiarti; 2013: 57)

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 1_____ 1..*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity.*
Asosiasi berarah/directed asosiasi 	Relasi antar kelas dengan makna kelas yang satu di gunakan oleh kelas yang lain, asosiasi biasanya juga di sertai 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*)

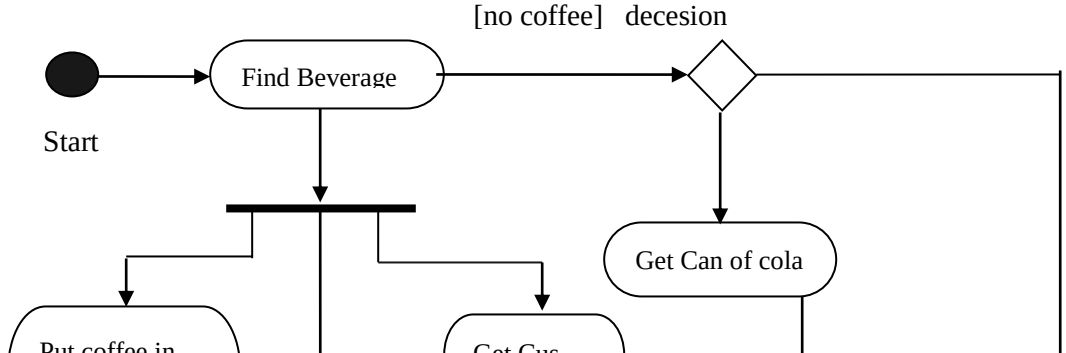
Sumber : (Yuni Sugiarti ; 2013)

II.7.3. *Activity* Diagram

Diagram aktivitas atau *activity* diagram menggambarkan *workflow* (aliran kerja) atau aktifitas dari sebuah sistem atau proses bisnis.

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. (Yuni Sugiarti; 2013: 75)



Gambar II.2 Activity Diagram

Sumber : (Yuni Sugiarti ; 2013)

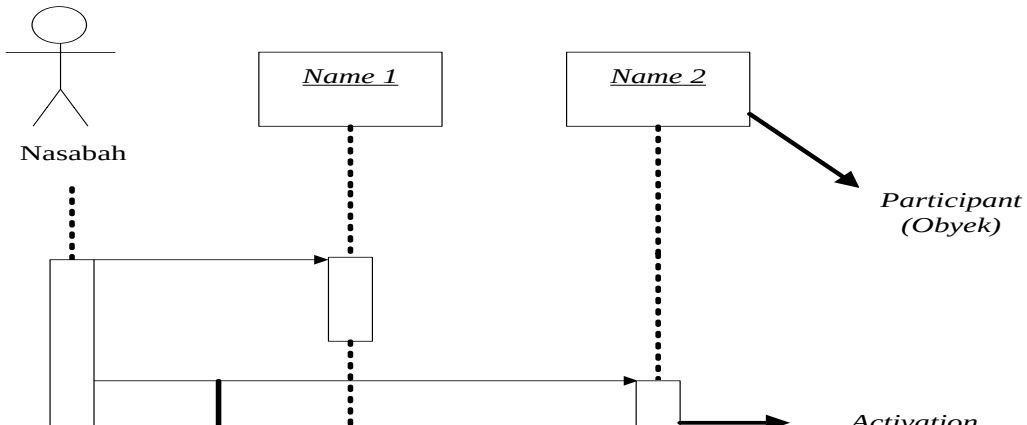
II.7.4. *Sequence* Diagram

Diagram sekuence menggambarkan kelakuan/ pelaku 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 diinstasiasi menjadi objek itu.

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

1. Pada diagram *sequence* terdapat garis hidup objek. Garis hidup objek adalah garis vertical yang mencerminkan eksistensi sebuah objek sepanjang periode waktu. Sebagian besar objek – objek yang tercakup dalam diagram interaksi akan eksis sepanjang durasi tertentu dari interaksi, sehingga objek – objek itu diletakkan dibagian atas diagram dengan garis hidup tergambar dari atas hingga bagian bawah diagram. Suatu objek lain dapat saja diciptakan, dalam hal ini garis hidup dimulai saat pesan *destroy*,
2. jika kasus ini terjadi, maka garis hidupnya juga berakhir.
3. Terdapat focus kendali (*Focus Of Control*), berupa empat persegi panjang ramping dan tinggi yang menampilkan aksi suatu objek secara langsung atau sepanjang sub ordinat. Puncak dari empat persegi panjang adalah permulaan aksi, bagian dasar adalah akhir dari suatu aksi. Pada diagram ini mungkin juga memperhatikan penyaringan (*nesting*) dan *focus* kendali yang disebabkan oleh proses rekursif dengan menumpuk *focus* kendali yang lain pada induknya. (Yuni Sugiarti; 2013: 70)

Berikut simbol – simbol yang ada pada sequence diagram.



Gambar II.3 Simbol *Squence*

Sumber : (Yuni Sugiarti ; 2013)