BAB II

TINJAUAN PUSTAKA

II.1. Perancangan

Perancangan adalah spesifikasi umum dan terinci dari pemecahan masalah berbasis komputer yang telah dipilih selama tahap analisis. Berdasarkan definisi tersebut dapat disimpulkan bahwa perancangan adalah kemampuan untuk membuat alternatif pemecahan masalah berbasis komputer selama tahap analisis. (Azhar Susanto, 2004:332)

II.2. Software

Software merupakan program-program komputer yang berguna untuk menjalankan atau mengoperasikan suatu pekerjaan sesuai dengan yang dikehendaki. Program tersebut ditulis dengan bahasa khusus yang dimengerti oleh komputer. Program dapat dianalogikan sebagai instruksi atau perintah-perintah untuk mengoperasikan atau menjalankan *hardware*. *Software* terdiri dari berbagai jenis, yaitu sistem operasi, program *utility* (bantuan), program aplikasi, dan program paket. (Sariadin, 2009:3)

II.3. Algoritma

Algoritma adalah urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis. Logis merupakan kunci dari sebuah algoritma. Algoritma dapat diekspresikan dalam bahasa manusia menggunakan presentasi

grafik atau simbol melalui sebuah *flow chart* (diagram alir) dan *pseudocode* yang menjembatani antara bahasa manusia dan bahasa pemrograman. (Sariadin, 2009:5)

II.4. Kriptografi

Kriptografi berasal dari bahasa yunani, menurut bahasa dibagi menjadi dua kripto dan graphia, kripto berarti *secret* (rahasia) dan graphia berarti writing (tulisan). Menurut terminologinya kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan ketika pesan dikirim dari suatu tempat ketempat yang lain.

II.4.1. Sejarah Kriptografi

Kriptografi mempunyai sejarah yang sangat menarik dan panjang. Kriptografi sudah digunakan 4000 tahun yang lalu yang diperkenalkan oleh orang-orang Mesir untuk mengirim pesan ke pasukan militer yang berada di lapangan dan supaya pesan tersebut tidak terbaca oleh pihak musuh walaupun kurir pembawa pesan tersebut tertangkap oleh musuh.

Pada zaman Romawi kuno dikisahkan pada suatu saat, ketika Julius Caesar ingin mengirimkan satu pesan rahasia kepada seorang Jendaral di medan perang. Pesan tersebut harus dikirimkan melalui seorang kurir, tetapi karena pesan tersebut mengandung rahasia, Julius Caesar memikirkan bagaimana mengatasinya yaitu dengan cara mengacak pesan tersebut menjadi suatu pesan yang tidak dapat dipahami oleh siapapun kecuali hanya dapat dipahami oleh Jenderalnya saja. Tentu sang Jenderal telah diberi tahu sebelumnya bagaimana cara membaca pesan yang teracak tersebut, karena telah mengetahui kuncinya. Yang dilakukan Julius

Caesar adalah mengganti semua susunan alphabet dari a, b, c, & yaitu a menjadi d, b menjadi e, c menjadi f dan seterusnya. Sehingga kalau Julius menuliskan kata "saya sekarang & vhndudqi & membacanya.

Dari ilustrasi tersebut, beberapa istilah *Cryptography* dipergunakan untuk menandai aktifitas-aktifitas rahasia dalam mengirim pesan. Apa yang dilakukan Julius Caesar dengan cara mengacak pesannya, kita sebut sebagai *encryption* dan pada saat Sang Jenderal merapikan pesan yang teracak itu, kita sebut dengan *decryption*. Pesan awal yang belum diacak dan yang telah dirapikan, kita sebut *plaintext* sedangkan pesan yang telah diacak, kita sebut *ciphertext*.

Huruf-huruf dengan bentuk tegak akan mempunyai lebar huruf yang lebih kecil dibandingkan dengan huruf-huruf yang melintang, sehingga dengan jumlah huruf yang sama, huruf yang berbentuk melintang akan memakan banyak tempat. Spasi antar huruf juga terlihat bervariasi pada huruf yang melintang daripada huruf tegak.

Pada perang dunia kedua, Jerman menggunakan enigma atau juga disebut dengan rotor yang digunakan Hitler untuk mengirim pesan ke tentaranya. Jerman sangat percaya pesan yang dikirim melalui enigma tidak terpecahkan kode-kode enkripsinya. Tapi anggapan itu keliru, setelah bertahun-tahun sekutu dapat memecahkan kode-kode tersebut terpecahkan, maka enigma yang digunakan pada perang dunia kedua, beberapa kali mengalami perubahan.

Enigma yang digunakana Jerman bisa mengenkripsikan satu pesan yang mempunyai 15 Miliyar kemungkinan untuk dapat mengedekripsikan satu pesan. Dan dari kemungkinan tersebut Jerman tidak percaya pesan yang dikirim melalui

enigma tersebut bisa dipecahkan. Tapi kenyataannya bisa didekripsikan oleh pihak sekutu. Cara kerja enigma akan dibahas pada bagian berikutnya.

Selama bertahun-tahun kriptografi menjadi bidang khusus yang hanya dipelajari oleh pihak militer, seperti agen keamanan Nasional Amerika (*National Security Agency*), Uni Soviet, Inggris, Perancis, Israel dan Negara-negara lainnya yang telah membelanjakan miliaran dolar untuk mengamankan komunikasi mereka dari pihak luar, tapi mereka selalu mempelajari kode-kode rahasia negara lain, dengan adanya persaingan ini maka kriptografi terus berkembang sesuai dengan perkembangan zaman.

Namun pada 30 tahun terakhir ini, kriptografi tidak hanya dimonopoli oleh pihak militer saja, hal yang sama juga dilakukan oleh individu-individu yang mengingikan pesan dan komunikasi mereka tidak diketahui oleh pihak lain, dan setiap individu berhak mengamankan informasi keluarganya, pekerjan bisnis, dan lainya. Apabila pada zaman sekarang ini persaingan yang begitu tinggi, mereka rela mengeluarkan sekian dolar hanya untuk menjaga *privacy* mereka.

II.4.2. Algoritma Kriptografi

Algoritma ditinjau dari asal usul kata, kata algoritma mempunyai sejarah yang menarik, kata ini muncul di dalam kamus webster sampai akhir tahun 1957 hanya menemukan kata algorism yang mempunyai arti proses perhitungan dengan bahasa arab. Algoritma berasal dari nama penulis buku arab yang terkenal yaitu Abu Ja'far Muhammad ibnu Musa al-Khuwarizmi (al- Khuwarizmi dibaca oleh

orang barat menjadi algorism). Kata algorism lambat laun berubah menjadi algorithm.

Defenisi terminologinya Algoritma adalah urutan langkah-langkah logis untuk penyelesaian masalah yang disusun secara sistimatis. Algoritma kriptografi merupakan langkah-langkah logis bagaimana menyembunyikan pesan dari orang-orang yang tidak berhak atas pesan tersebut.

1. Enkripsi : enkripsi merupakan hal yang sangat penting dalam kriptografi yang merupakan pengamanan data yang dikirimkan terjaga rahasiannya. Pesan asli disebut *plaintext* yang dirubah jadi kode-kode yang tidak dimengerti. Enkripsi bisa diartikan dengan chipper atau kode. Sama halnya dengan kita tidak mengerti akan sebuah kata, maka kita akan melihatnya didalam kamus atau daftar istilah-istialah. Beda halnya dengan enkripisi, untuk merubah *plaintext* ke bentuk *chippertext* kita menggunakan algoritma yang dapat mengkodekan data yang kita inginkan.
2. Dekripsi : dekripsi merupakan kebalikan dari enkripsi dikembalikan berbentuk asalnya (*Plaintex*t) disebut dengan dekripsi pesan. Algoritma yang digunakan untuk enkripsi dengan dekripsi tentu berbeda dengan yang digunakan untuk keinginan.
3. Kunci : kunci yang dimaksud disini adalah kunci yang dipakai untuk melakukan enkripsi dan dekripsi, kunci terbagi menjadi 2 bagian kunci pribadi (*private key*) dan kunci umum (*public key*).

Keamanan dari algoritma kriptografi tergantung dari bagaimana suatu algoritma itu bekerja, maka algoritma semacam ini disebut dengan algoritma terbatas.

Algoritma terbatas merupakan suatu algoritma yang dipakai sekelompok orang untuk merahasiakan pesan yang dikirimnya. Jika salah satu dari anggota kelompok itu keluar dari kelompoknya maka, algoritma yang dipakai diganti dengan yang baru, jika tidak hal ini bisa menjadi masalah dikemudian hari.

Keamanan dari kriptografi modern hanya dengan merahasiakan kunci yang dimiliki dari orang lain, tanpa harus merahasiakan algoritma itu sendiri,kunci befungsi sama dengan password. Jika keseluruhan dari keamanan algoritma tergantung pada kunci yang dipakai maka, algoritma ini bisa dipublikasikan dan dianalisis oleh orang lain. Jika algoritma ini bisa dipublikasikan dan bisa dipecahkan dalam waktu singkat oleh orang lain maka, algoritma tersebut belum aman untuk digunakan. Pada pembahasan berikutnya akan dijelaskan berbagai macam algoritma kriptografi yang pernah ada.

II.4.3. Macam-macam Algoritma Kriptografi

Algoritma kriptografi dibagi menjadi tiga bagian berdasarkan dari kunci yang dipakainya :

1. Algoritma Simetri (menggunakan satu kunci untuk enkripsi dan dekripsinya)
2. Algoritma Asismetri (menggunakan kunci yang berbeda untuk enkripsi dan dekrispi)
3. *Hash Function*

II.4.3.1. Algoritma Simetri

Algoritma ini juga sering disebut dengan algoritma klasik, karena memakai kunci yang sama untuk kegiatan enkripsi dan dekripsinya. Algoritma ini sudah ada lebih 4000 tahun yang lalu. Mengirim pesan dengan menggunakan algoritma ini, sipenerima pesan harus diberitahu kunci dari pesan tersebut agar bisa mendekripsi pesan yang dikirim. Keamanan dari pesan yang menggunakan algoritma ini tergantung pada kunci, jika kunci tersebut diketahui oleh orang lain maka, orang tersebut bisa melakukan enkripsi dan dekripsi terhadap pesan tersebut. Algoritma yang memakai kunci simetri diantaranya adalah :

a. *Data encryption standard* (DES)
b. RC2, RC4, RC5, RC6
c. *International Data Encryption Algoritma* (IDEA)
d. *Advanced Encryption Standard* (AES)
e. *One Time Pad* (OTP)
f. A5
g. Dan lain sebgainya.

II.4.3.2. Algoritma Asimetri

Algoritma asimetri sering juga disebut dengan algoritma kunci publik, dengan arti kata kunci yang digunakan untuk melakukan enkripsi dan dekripsinya berbeda. Pada algoritma asimetri kunci terbagi menjadi dua bagian:

1. Kunci umum (*public key*): kunci yang boleh semua orang tahu (dipublikasikan)

2. Kunci pribadi (*private key*): kunci yang dirahasiakan (hanya boleh diketahui oleh satu orang)

Kunci-kunci tersebut saling berhubungan satu dengan yang lainnya. Dengan kunci publik orang dapat mengenkripsi pesan tapi tidak bisa mendekripsikan, hanya orang yang memiliki kunci pribadi yang dapat mendekripsi pesan tersebut. Algoritma asimentris bisa melakukan pengiriman pesan lebih aman dari pada algoritma simetris.

Contoh : Bob mengirim pesan ke Alice menggunakan asimetris, hal yang harus dilakukan adalah :

a. Bob memberitahukan kunci publiknya ke Alice
b. Alice mengenkripsi pesan dengan menggunakan kunci publik Bob
c. Bob mendekripsi pesan dari Alice dengan kunci pribadinya
d. Dan begitu juga sebaliknya jika bob ingin mengirim pesan ke Alice.

Algoritma yang memakai kunci publik diantaranya adalah :

a. *Digital signature algorithm* (DSA)
b. RSA
c. *Diffie-Hellman* (DH)
d. *Elliptic curve Cryptography* (ECC)
e. Dan lain sebagainya

II.4.3.3. Fungsi Hash

Fungsi hash sering di sebut dengan fungsi hash satu arah *(one-way-function),message digest*, *fingerprint*, fungsi kompresi dan *message*

authentication code (MAC), hal ini merupakan suatu fungsi matematika yang mengambil input panjang variabel dan mengubahnya ke dalam urutan biner dengan panjang yang tetap.

II.4.4. Kriptografi Klasik

Kriptografi klassik merupakan suatu algoritma yang menggunakan satu kunci untuk mengamankan data, teknik ini sudah diguanakan beberapa abad yang lalu. Dua teknik dasar yang biasa digunakan pada algoritma jenis ini, diantaranya adalah :

1. Teknik subtitusi : pengantian setiap karakter *plaintext* dengan karakter lain
2. Teknik transposisi (permutasi) : teknik ini menggunakan permutasi karakter.

II.4.4.1. Teknik Subtitusi

Substitusi merupakan pergantian setiap karakter dari *plaintext* ke karakter lainnya. Jenis-jenis dari teknik ini adalah:

1. *Caesar Cipher*
2. *Playfair Cipher*
3. *Shift Cipher*
4. *Hill Cipher*
5. *Vigenere Cipher*

II.4.4.2. Teknik Transposisi

Teknik ini menggunakan permutasi karakter, yang mana dengan menggunakan teknik ini pesan yang asli tidak dapat dibaca kecuali memiliki kunci untuk mengembalikan pesan tersebut kebentuk semula atau disebut dengan dekripsi.

II.4.5. Kriptografi Modern

Kriptografi modern merupakan suatu algoritma yang digunakan pada saat sekarang ini, yang mana kriptografi modern mempunyai kerumitan yang sangat komplek, karena dalam pengoperasinya menggunakan komputer.

II.4.5.1. Macam-macam Algoritma Kriptografi Modern

Algoritma dari kriptografi modern terdiri atas dua bagian yaitu:

1. Simetri Algoritma

Simetri Algoritma adalah algoritma yang menggunakan kunci yang sama pada enkripsi dan dekripsinya.

Aplikasi dari algoritma simetri yang dipergunakan oleh beberapa algoritma dibawah ini :

a. *Data encryption standard* (DES)
b. *Advanced Encryption Standard*(AES)
c. *International Data Encryption Algoritma* (IDEA)
d. A5
e. RC4

2. Asimetri Algoritma

Kunci asimetris adalah pasangan kunci kriptografi yang salah satunya digunakan untuk proses enkripsi dan yang satunya lagi untuk dekripsi.

Asimetri Algoritma terbagi menjadi beberapa algoritma antara lain :

a. RSA
b. Knapsack Merkle-Hellman
c. Algoritma ElGamal
d. Fungsi Hash Satu Arah
e. Algoritma *DES-like Message Digest Computation* (DMDC)
f. MD-5
g. *Elliptic Curve Cryptosystem* (ECC)

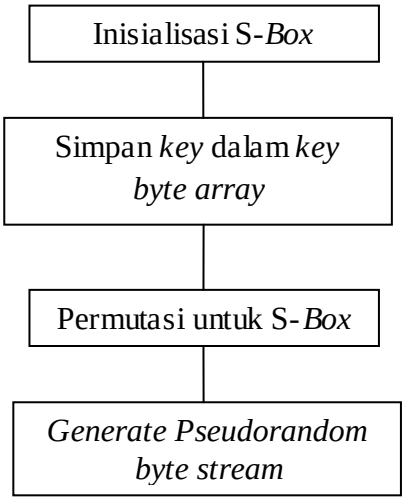
(Ariyus, 2006)

II.5. Algoritma RC4

Algoritma RC4 dikembangkan oleh Ronald Rivest pada tahun 1984 merupakan system sandi strean yang paling banyak digunakan misalnya pada protocol SSL/TSL, (Stalling 2006). RC4 merupakan system sandi stream berorientasi *byte*. Masukan algoritma enkripsi RC4 merupakan sebuah *byte*, kemudian dilakukan operasi XOR dengan sebuah *byte* kunci, dan menghasilkan sebuah *byte* sandi. (sumber : Sadikin, Rifki, 2012, ***Kriptografi***, Penerbit Andi, Yogyakarta.)

Proses enkripsi dan dekripsi RC4 mempunyai proses yang sama sehingga hanya ada satu fungsi yang dijalankan untuk menjalankan kedua proses tersebut.

Berikut ini akan diberikan sebuah bagan yang menggambarkan rangkaian proses yang dijalankan untuk mengenripsi atau mendekripsi pada RC4.



Gambar II.1 Enkripsi dan Dekripsi RC4

Untuk lebih jelasnya tahapan-tahapan enkripsi dan dekripsi RC4 adalah sebagai berikut.

a. Pengguna memasukkan *secret key*
b. Inisialisasi awal S-*Box* berdasarkan indeks
c. Simpan *secret key* yang telah dimasukkan *user* kedalam *array* 256 *byte*
d. Bangkitkan nilai *pseudorandom* berdasarkan nilai *key sequence*
e. Proses permutasi nilai dalam S-*Box* selama 256 kali
f. Bangkitkan nilai *pseudorandom key byte stream* berdasarkan indeks dan nilai S-*Box*
g. Lakukan opersai XOR antara *plaintext/ciphertext* dan *pseudorandom key*.

(sumber : Jurnal Siti Mariyam, 2015)

II.6. Enkripsi

Enkripsi adalah sebuah proses yang melakukan perubahan sebuah kode dari yang bisa dimengerti menjadi sebuah kode yang tidak bisa dimengerti (tidak terbaca). Enkripsi dibentuk berdasarkan suatu algoritma yang akan mengacak suatu informasi menjadi bentuk yang tidak bisa dibaca atau tidak bisa dilihat. (Andi, 2003)

II.7. Dekripsi

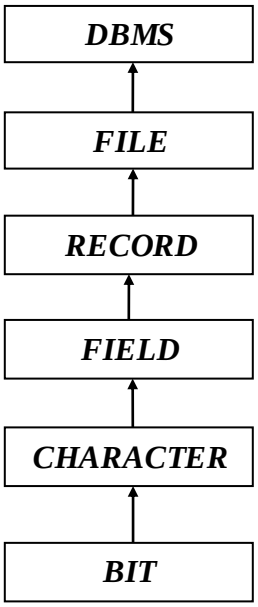
Dekripsi adalah proses dengan algoritma yang sama untuk mengembalikan informasi teracak menjadi bentuk aslinya. Algoritma yang digunakan harus terdiri dari susunan prosedur yang direncanakan secara hati-hati yang harus secara efektif menghasilkan sebuah bentuk ter-enkripsi yang tidak bisa dikembalikan oleh seseorang, bahkan sekalipun mereka memiliki algoritma yang sama. (Andi, 2003)

II.8. File

File adalah kumpulan data atau informasi yang mempunyai nama. Hampir semua informasi dalam komputer harus disimpan dalam bentuk file. Ada beberapa jenis file, seperti file data, file teks file program, dan sebagainya. Setiap jenis file menyimpan jenis informasi yang berbeda. Sebagai contoh file program menyimpan program sedang file teks menyimpan teks.

Setiap pekerjaan atau operasional pada sistem komputer harus menggunakan file. File adalah kumpulan dari beberapa record yang mempunyai

kesatuan untuk dapat difungsikan. Perhatikan hirarki *database management system* (DBMS) berikut ini. **(**Sariadin, 2009:161)



Gambar II.2 Hirarki *DBMS*

II.9. Eclipse

Eclipse adalah sebuah IDE(*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua platform (*platform-independent).* Berikut ini adalah sifat dari Eclipse :

1. *Multi-platform*: Target sistem operasi Eclipse adalah *Microsoft Windows, Linux,Solaris, AIX, HP-UXdan Mac OS X.*
2. *Mulit-language:* Eclipse dikembangkan dengan bahasa pemrograman *Java*, akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemrograman lain seperti *C/C++, Cobol, Python, Perl, PHP,* dan lain sebagainya.

3. *Multi-role:* Selain sebagai IDE untuk pengembangan aplikasi. Eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak seperti dokumentasi, pengujian perangkat lunak, pengembangan web, dan lain sebagainya.

Sejak versi 3.0, Eclipse pada dasarnya merupakan sebuah *kernel*. Apa yang dapat digunakan di dalam Eclipse sebenarnya adalah fungsi dari *plug-in* yang sudah dipasang (*diinstal*). Ini merupakan basis dari Eclipse yang dinamakan *Rich Client Platform*(RCP). Berikut ini adalah komponen yang membentuk RCP:

Eclipse selalu dilengkapi dengan JDT(*Java Development Tools*), *plug-in* yang membuat Eclipse kompatibel untuk mengembangkan program *Java*, dan PDE(*Plug-in Development Environment*) untuk mengembangkan *plug-in* baru. (Sumber : Wina Noviani Fatimah, ST : 2011 : 2)

II.10. *Android*

Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis *linux* yang mencakup sistem operasi, *middleware* dan aplikasi. *Android* menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Awalnya *Google inc* membeli *android inc* yang merupakan pendatang baru yang membuat piranti lunak untuk ponsel/*smartphone*. Kemudian untuk mengembangkan *android*, dibentuklah *Open Handesti Alliance*, konsorium 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 Handest 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. (Safaat H, Nazruddin, 2012)

Aplikasi *Android* ditulis dalam bahasa pemrograman *java*, yaitu kode *java* yang terkompilasi bersama-sama dengan data dan *file-file* sumber yang dibutuhkan oleh aplikasi yang digabungkan oleh *app tools* menjadi paket aplikasi *Android*, sebuah file yang ditandai dengan akhiran *.apk* file inilah yang didistribusikan sebagai aplikasi dan diinstal pada *handset Android*. (Sumber : Ahmad, 2015 :191)

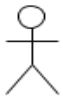
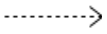
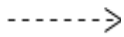
II.11. *Unified Modeling Language* (UML)

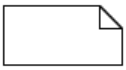
UML yang merupakan singkatan dari *Unified Modelling Language* adalah sekumpulan pemodelan konvensi yang digunakan untuk menentukan atau menggambarkan sebuah sistem perangkat lunak dalam kaitannya dengan objek. UML dapat juga diartikan sebuah bahasa grafik standar yang digunakan untuk memodelkan perangkat lunak berbasis objek. UML pertama kali dikembangkan pada pertengahaan tahun 1990an dengan kerjasama antara James Rumbaugh, Grady Booch dan Ivar Jacobson, yang masing-masing telah mengembangkan notasi mereka sendiri di awal tahun 1990an. (Lethbride dan Leganiere, 2009:11)

II.11.1. *Use Case Diagram*

Use case diagram, adalah sebuah gambaran dari fungsi sistem yang dipandang dari sudut pandang pemakai. *Actor* adalah segala sesuatu yang perlu berinteraksi dengan sistem untuk pertukaran informasi. *System boundary* menunjukkan cakupan dari sistem yang dibuat dan fungsi dari sistem tersebut. (Lethbride dan Leganiere, 2009:11)

Tabel II.1. Simbol *Use Case Diagram*

O	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
3		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
4		*Include*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.
5		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
6		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.

8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi

Gambar II.3. Simbol-simbol yang ada pada Use Case Diagram

(Sumber :Lethbride dan Leganiere, 2009:11)

II.11.2. *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. Menggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses. Dipakai pada *business modeling* untuk memperlihatkan

urutan aktifitas proses bisnis. Struktur diagram ini mirip *flowchart* atau Data *Flow* Diagram pada perancangan terstruktur. *Activity Diagram* dibuat berdasarkan sebuah atau beberapa *use case* pada *use case* diagram. (Lethbride dan Leganiere, 2009:13)

Tabel II.2. Simbol *Activity Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1		*Actifity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain
2		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi
3		*Initial Node*	Bagaimana objek dibentuk atau diawali.
4		*Actifity Final Node*	Bagaimana objek dibentuk dan dihancurkan
5		*Fork Node*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran

Gambar II.4. Simbol-simbol yang ada pada *Activity Diagram*

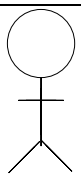
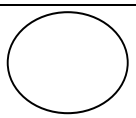
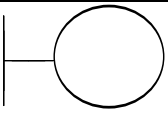
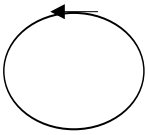
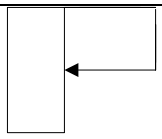
(Sumber :Lethbride dan Leganiere, 2009:15)

II.11.3. *Sequence diagram*

Sequence diagram menambahkan dimensi waktu pada interaksi diantara obyek. Pada diagram ini participant diletakkan di atas dan waktu ditunjukkan dari atas ke bawah. *Life line participant* diurutkan dari setiap paricipant. Kotak kecil pada life line menyatakan *activation* : yaitu menjalankan salah satu operation dari participant. Satate bisa ditambahkan dengan menempatkannya sepanjang life line. *Message* (sederhana, synchronous atau *asynchroneous*) adalah tanda panah yang

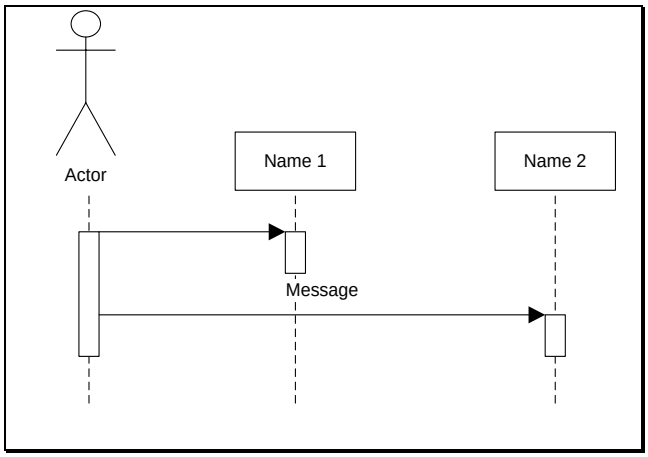
menghubungkan suatu *life line* ke *life line* yang lain. Lokasi *life line* dalam dimensi vertikal mewakili urutan waktu dalam sequence diagram. Message yang pertama terjadi adalah yang paling dekat dengan bagian atas diagram dan yang terjadi belakangan adalah yang dekat dengan bagian bawah. Pada beberapa sistem, operasi bisa dilakukan kepada dirinya sendiri. Hal ini disebut dengan rekursif. Untuk melukiskannya digunakan anak panah dar activation kembali ke dirinya sendiri, dan sebuah kotak kecil diletakkan pada bagian atas dari *activation*.

Tabel II.3 Simbol *Sequence Diagram*

Gambar	Keterangan
	*Actor* dapat berupa manusia, sistem atau *device* yang memliki peranan dalam keberhasilan operasi dari sistem.
	*Entity Class* merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data
	*Boundary Class* berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih actor dengan sistem.
	*Control Class* suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas.
	Message simbol mengirim pesan antar *class*.
	*Recursive* menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.

	*Activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi .
	Lifeline yaitu garis titik titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

(Sumber : Gellysa Urva dkk ; 2015 : 95)



Gambar II.5. Simbol-simbol yang ada pada *sequence diagram*

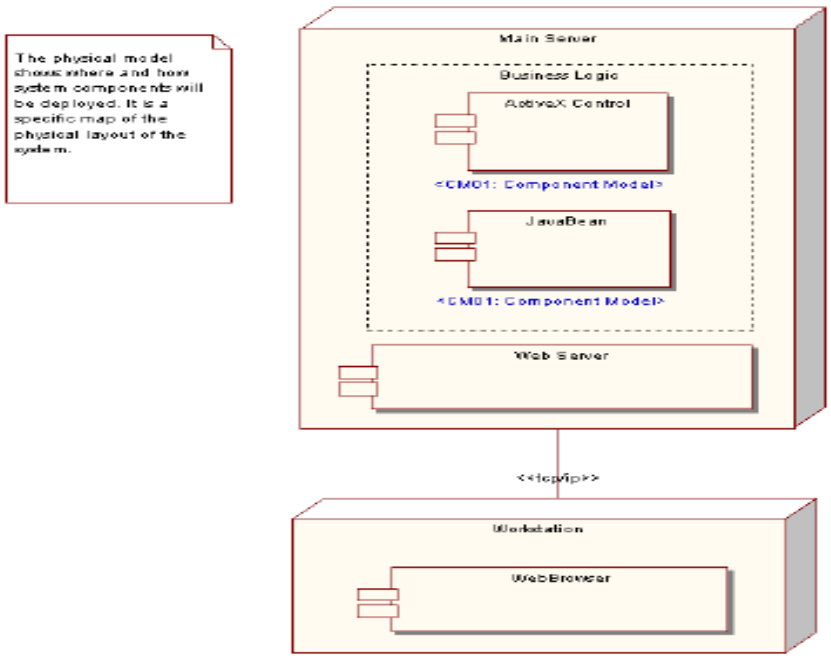
(Sumber :Agus Putranto, 2009:14)

II.11.4. *Deployment diagram*

Deployment diagram menggambarkan sumber fisik dalam sistem, termasuk node, komponen dan koneksi (model implementasi sistem yang statistik). Dalam hal ini meliputi topologi *hardware* yang dipakai sistem.

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak

(pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal. Sebuah *node* adalah server, *workstation*, atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.



Gambar II.6. Simbol-simbol yang ada pada *Deployment* Diagram

(Sumber :Agus Putranto:2009:9)