BAB II

TINJAUAN PUSTAKA

II.1. Perancangan

Untuk membuat tampilan yang menarik memang tidak mudah dilakukan. Seorang perancang tampilan selain harus mempunyai jiwa seni yang memadai, ia juga harus mengerti selera pengguna secara umum. Hal lain yang pelu disadari oleh seorang perancang tampilan adalah bahwa ia harus bisa meyakinkan pemrogramnya bahwa apa yang ia bayangkan dapat diwujudkan dengan peranti bantu yang tersedia [1].

Perancangan merupakan proses pengolahan hasil analisis perangkat lunak menjadi rencana pengembangan perangkat lunak dan batasan-batasan perangkat lunak atau masalah yang mungkin dihadapi dalam pengembangan perangkat lunak. Perancangan yang dilakukan meliputi perancangan arsitektur, perancangan modul, dan perancangan antarmuka. [2].

Bagi perancang antarmuka, hal yang sangat penting untuk ia perhatikan adalah mendokumentasikan semua pekerjaan yang dilakukan. Dokumentasi rancangan dapat dikerjakan atau dilakukan dengan beberapa cara :

1. Membuat sketsa pada kertas
2. Menggunakan peranti purwarupa GUI
3. Menuliskan keterangan yang menjelaskan tentang kaitan antara jendela.
4. Menggunakan peranti bantu CASE (*Computer Aided Software Engineering*).

Cara kedua dan keempat tidak selalu dapat diterapkan, karena peranti tersebut biasanya harus dibeli dan seringkali cukup mahal. Cara ini kebanyakan diterapkan pada pembuatan antarmuka grafis untuk suatu jenis pekerjaan berskala besar.

II.1.1. Cara Pendekatan

Sebuah program aplikasi pastilah ditujukan kepada pengguna, yang utama, bukan perancangan program aplikasi tersebut. Program aplikasi pada dasarnya dapat dikelompokkan dalam dua kategori besar, yakni program aplikasi untuk keperluan khusus dengan pengguna yang khusus pula dan program aplikasi yang akan digunakan oleh pengguna umum, yang juga sering dikenal dengan sebutan public software. Karena perbedaan pada calon pengguna, maka perancang program antarmuka perlu memperhatikan hal ini [1].

Pada kelompok pertama, yakni pada program aplikasi untuk keperluan khusus, misalnya program aplikasi untuk inventori gudang, pengelolaan data akademis mahasiswa, pelayanan reservasi hotel, dan program-program aplikasi yang serupa, kelompok calon pengguna yang akan memanfaatkan program aplikasi tersebut dapat dengan mudah diperkirakan, baik dalam hal keahlian pengguna maupun ragam antarmuka yang akan digunakan. Untuk kelompok ini ada satu pendekatan yang dapat dilakukan, yakni pendekatan yang disebut dengan pendekatan perancangan berpusat ke pengguna (*user centered design approach*). Cara pendekatan ini berbeda pendekatan perancangan oleh pengguna (*user design approach*).

Pendekatan perancangan berpusat ke pengguna adalah perancangan antarmuka yang melibatkan pengguna. Pelibatan pengguna di sini tidak diartikan bahwa pengguna harus ikut memikirkan bagaimana implementasinya nanti, tetapi pengguna diajak untuk aktif berpendapat ketika perancangan antarmuka sedang menggambar wajah antarmuka yang mereka inginkan. Dengan kata lain, perancangan dan pengguna duduk bersama-sama untuk merancang wajah antarmuka yang diinginkan pengguna. Pengguna menyampaikan keinginannya. Sementara perancangan menggambar keinginan pengguna tersebut sambil menjelaskan keuntungan dan kerugian wajah antarmuka yang diinginkan oleh pengguna, seolah-olah sudah mempunyai gambaran nyata tentang antarmuka yang nanti akan mereka gunakan [1].

Pada perancangan oleh pengguna, pengguna sendirilah yang merancang wajah antarmuka yang diinginkan. Di satu sisi, cara ini akan mempercepat proses pengimplementasian modul antarmuka. Tetapi di sisi yang lain, hal ini justru sangat memberatkan pemrogram karena apa yang diinginkan pengguna belum tentu dapa diimplementasikan dengan mudah, atau bahkan tidak dapat dikerjakan dengan menggunakan peranti bantu yang ada.

Perancang program aplikasi yang dimasukkan dalam kelompok kedua, aau public *software*, perlua menganggap bahwa program aplikasi tersebut akan digunakan oleh pengguna dengan berbagai tingkat kepandaian dan karakteristik yang sangat beragam. Di satu sisi keadaan ini dapat ia gunakan untuk memaksa pengguna menggunakan antarmuka yang ia buat, tetapi pada sisi lain pemaksaan itu akan berakibat bahwa program aplikasinya menjadi tidak banyak

penggunanya. Satu kunci penting dalam pembuatan modul antarmuka untuk program-program aplikasi pada kelompok ini adalah dengan melakukan customization. Dengan customization pengguna dapat menggunakan program aplikasi dengan wajah antarmuka yang sesuai dengan selera masing-masing pengguna.

Salah satu contoh dari adanya kemampuan yang dimiliki oleh sebuah program aplikasi atau sistem operasi yang dapat disesuaikan dengan karakteristik pengguna adalah pengaturan desktop pada OS X versi 10.5 favoritnya, sehingga pengguna dapat mengubahnya sesuai keinginan justru akan membuat mata pengguna itu sakit, dikarenakan mata harus melakukan akomomasi maksimum terus menerus untuk menyesuaikan dengan warna tampilan yang ada.

Selain cara pendekatan yang dijelaskan di atas, Anda yang terbiasa menulis program-program aplikasi mungkin mempunyai cara khusus untuk berhadapan dengan pengguna. Tetapi perlu Anda ingat bahwa apapun cara yang Anda gunakan, Anda tetap harus mempunyai pedoman bahwa pada akhirnya program itu bukan untuk Anda sendiri, tetapi akan digunakan oleh orang lain. Dengan kata lain, jangan pernah mengabaikan pendapat (calon) pengguna program aplikasi Anda.[1]

II.1.2. Prinsip Dan Petunjuk Perancangan

Antarmuka pengguna secara alamiah terbagi menjadi empat komponen model pengguna, bahasa perintah, umpan balik, dan penampilan informasi. Model pengguna merupakan dasar dari tiga komponen yang lain.

Model mental pengguna merupakan model konseptual yang dimiliki oleh pengguna ketika ia menggunakan sebuah sistem atau program aplikasi. Model ini memungkinkan seorang pengguna untuk mengembangkan pemahaman mendasar tentang bagian yang yang dikerjakan oleh program, bahkan oleh pengguna yang sama sekali tidak mengetahui teknologi komputer. Dengan pertolongan model itu pengguna dapat mengantisipasi pengaruh suatu tindakan yang dilakukan dan dapat memilih strategi yang cocok untuk mengoperasikan program tersebut. Model pengguna dapat berupa suatu simulasi tentang keadaan yang sebenarnya dalam dunia nyata, sehingga ia tidak perlu mengembangkannya sendiri dari awal.

Setelah pengguna mengetahui dan memahami model yang di inginkan, dia memerlukan peranti untuk memanipulasi model itu. Peranti pemanipulasian model ini sering disebut dengan bahasa perintah (*command language*), yang sekaligus merupakan komponen kedua dari antarmuka pengguna. Idealnya program komputer kita mempunyai bahasa perintah yang alami, sehingga model pengguna dengan cepat dapat dioperasionalkan. [1]

Komponen ketiga adalah umpan balik. Umpan balik di sini diartikan sebagai kemampuan sebuah program yang membantu pengguna untuk mengoperasikan program itu sendiri. Umpan balik dapat berbentuk pesan penjelasan, pesan penerimaan perintah, indikasi adanya obyek terpilih, dan penampilan karakter yang diketikkan lewat papan ketik. Beberapa bentuk umpan balik terutama ditujukan kepada pengguna pengguna yang belum berpengalaman dalam menjalankan program sebuah aplikasi. Umpan balik dapat digunakan untuk

member keyakinan bahwa program telah menerima perintah pengguna dan dapat memahami maksud perintah tersebut.

Komponen keempa adalah tampilan informasi. Komponen ini digunakan untuk menunjukkan status informasi atau program ketika pengguna melakukan suatu tindakan. Pada bagian ini perancang harus menampilkan pesan-pesan tersebut seefektif mungkin sehingga mudah dipahami oleh pengguna. Setelah memahami beberapa prinsip dalam perancangan antarmuka pengguna. Pada bagian berikut ini akan diberikan petunjuk singkat tentang perancangan anarmuka yang akan Anda lakukan sebagai seorang perancang tampilan.

II.1.3. Urutan Perancangan

Perancangan dialog, seperti halnya perancangan sistem yang lain, harus dikerjakan secara aas ke bawah. Proses perancangannya dapat dikerjakan secara bertahap sampai rancangan yang diinginkan terbentuk, yaitu sebagai berikut

1. Pemilihan ragam dialog

Untuk suatu tugas tertentu, pilihlah ragam dialog yang menurut perkiraan cocok untuk tugas tersebut. Ragam dialog dapat dipilih dari sejumlah ragam dialog yang telah dijelaskan pada bab-bab sebelumnya. Pemilihan ragam dialog dipengaruhi oleh karakterisiktik populasi pengguna, tipa dialog yang diperlukan, dan kendala teknologi yang ada untuk mengimplementasikan ragam dialog tersebut.[1]

2. Perancangan Struktur Dialog

Tahap kedua adalah melakukan analisis tugas dan menentukan model pengguna dari tugas tersebut untuk membentuk struktur dialog yang sesuai.

Dalam tahap ini pengguna sebaiknya banyak dilibatkan, sehingga pengguna langsung mendapatkan umpan balik dari diskusi yang terjadi. Pada tahap ini suatu purwarupa dialog seringkali dibuat untuk memberik gambarang yang lebih jelas kepada calon pengguna.

3. Perancangan format pesan

Pada tahap ini tata letak tampilan dan keterangan tekstual secara terinci harus mendapat perhatian lebih. Selain itu, kebutuhan data masukan yang mengharuskan pengguna untuk memasukkan data ke dalam komputer juga harus dipertimbangkan dari segi efisiensinya. Salah satu contohnya adalah dengan mengurangi pengetikan yang tidak perlu dengan cara mengefektifkan pengguna tombol.

II.2. Algoritma Kriptografi

Ditinjau dari asal usulnya, kata algoritma mempunyai sejarah yang menarik. Kata ini muncul di dalam kamus Webster sampai akhir tahun 1957. Kata algorisma mempunyai arti proses perhitungan dalam 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 sebagai algorism). Kata algorism lambat laun berubah menjadi algorithm.

Definisi terminologi algoritma adalah urutan langkah-langkah logis untuk menyelesaikan masalah yang disusun secara sistematis. Algoritma kriptografi merupakan langkah-langkah logis bagaimana menyembunyikan pesan dari orang-

orang yang tidak berhak atas pesan tersebut. Algoritma kriptografi terdiri dari tiga fungsi dasar, yaitu :

1. Enkripsi merupakan hal yang sangat penting dalam kriptografi, merupakan pengamanan data yang dikirimkan agar terjaga kerahasiannya. Pesan asli disebut plaintext, yang diubah menjadi kode-kode. Sama halnya dengan tidak mengerti akan sebuah kata maka akan melihatnya di dalam kamus atau daftar istilah. Bedan halnya dengan enkripsi, untuk mengubah teks asli ke bentuk teks kode menggunakan algoritma yang dapat mengkodekan data yang diinginkan.
2. Deskripsi merupakan kebalikan dari enkripsi. Pesan yang telah dienkripsi dikembalikan ke bentuk asalnya teks asli, disebut dengan enkripsi pesan. Algoritma yang digunakan untuk deskripsi tentu berbeda dengan algoritma yang digunakan untuk enkripsi.
3. Kunci yang dimaksud di sini adalah kunci yang dipakai untuk melakukan enkripsi dan deskripsi. Kunci terbagi menjadi dua bagian, kunci rahasia (*private key*) dan kunci umum (*public key*).
4. Keamanan dari algoritma kriptografi tergantung pada bagaimana algoritma itu bekerja. Oleh sebab itu algoritma semacam ini disebut dengan algoritma terbatas. Algoritma terbatas merupakan algoritma yang dipakai sekelompok orang untuk merahasiakan pesan yang mereka kirim. Jika salah satu dari anggota kelompok itu keluar dari kelompoknya amak algoritma yang dipakai diganti dengan yang baru. Jika tidak maka hal itu bisa menjadi masalah di kemudian hari.

II.3. Macam-macam algoritma Kriptografi

Algoritma kriptografi dibagi menjadi tiga bagian berdasarkan kunci yang dipakainya, sebagai berikut :

1. Algoritma Simetri.

   Pada sistem kriptografi kunci simetris, kunci untuk enkripsi sama dengan kunci untuk dekripsi, oleh karena itulah dinamakan kriptografi simetris. Keamanan sistem kriptografi simetris terletak pada kerahasiaan kuncinya. Kriptografi simetris merupakan satu-satuanya jenis kriptografi yang dikenal dalam catatan sejarah hingga tahun 1976.

2. Algoritma Asimetri.

   Algoritma asimetris menggunakan dua jenis kunci, yaitu kunci publik dan kunci rahasia. Nama lainnya adalah kriptografi kunci publik, sebab kunci untuk enkripsi tidak rahasia dan dapat diketahui oleh siapapun, sementara kunci untuk dekripsi hanya diketahui oleh penerima pesan.

3. *Hash Function*.

   Algoritma ini sering disebut dengan algoritma klasik karena memakai kunci yang sama untuk kegiatan enkripsi dan dekripsi. Algoritma ini sudah ada sejak lebih dari 4000 tahun yang lalu. Bila mengirim pesan dengan menggunakan algoritma ini, si penerima pesan harus diberitahu kunci dari pesan tersebut agar bisa mendekripsi pesan yang dikirim.

Dari ketiga kunci yang telah dijelaskan diatas, kunci tersebut bersifat rahasia baik itu kunci enkripsi maupun kunci dekripsi. Kunci ini dapat digunakan sebagai

private maupun publik, dan kunci ini dapat digunakan secara bersamaan antara kunci enkripsi dan kunci dekripsi.

II.4. RC5 (*Rivest Code 5*)

Algoritma enkripsi RC5 didesain oleh Profesor Ronald Rivest dan pertama kali dipublikasikan pada Desember 1994. Sejak publikasinya RC5 telah menarik perhatian banyak peneliti dalam bidang kriptografi dalam rangka menguji tingkat keamanan yang ditawarkan oleh algoritma RC5 (*RSA Laboratory Technical Report TR-602*). [3].

Algoritma RC5 merupakan metode enkripsi menggunakan metode simetrik dan pengolahan dalam bentuk blok *chiper*, jadi kata kunci yang sama digunakan untuk proses enkripsi dan dekripsi. Parameter-parameter yang digunakan dalam RC-5 adalah sebagai berikut :

1. Jumlah putaran ini disimbolkan dengan *r* yang merupakan parameter untuk rotasi dengan nilai 0, 1, 2, …… 255.
2. Jumlah *word* dalam bit disimbolkan dengan *w*. Nilai bit yang di *support* adalah 16 bit, 32 bit, dan 64 bit.
3. Kata kunci (*key word*) Variable ini disimbolkan dengan *b* dengan range 0, 1, 2, …. 255. *Key word* ini dikembangkan menjadi *array S* yang digunakan sebagai *key* pada proses untuk enkripsi dan dekripsi.

Pada dasarnya algoritma RC5 mempunyai syarat-syarat dalam mendesain dengan sedemikian rupa untuk memenuhi syarat-syarat tersebut, syarat-syarat yang dimaksud adalah sebagai berikut :

1. RC5 harus merupakan blok chiper simetris. Kunci rahasia yang sama dalam kriptografi digunakan dalam enkripsi dan dekripsi. Teks awal dan teks terenkripsi memiliki panjang yang sudah ditentukan dalam masing-masing blok.
2. RC5 harus cocok untuk *hardware* ataupun *software*. Hal tersebut berarti RC5 hanya akan menggunakan operasi perhitungan primitif yang sering kali ditemukan pada mikroprosessor.
3. RC5 harus cepat, hal tersebut lebih kurang dikarenakan RC5 merupakan algoritma *wordoriented* dengan kata lain pada operasi komputasi dasar yang digunakan harus dapat memproses data *word* secara penuh dalam satu waktu.
4. RC5 harus dapat beradaptasi pada berbagai panjang data word. Semisal pada prosessor 64 bit, panjang data *word* yang digunakan lebih panjang daripada prosesor 32 bit. RC5 harus dapat memanfaatkan hal tersebut, oleh karenanya RC5 memiliki parameter *w* yang yang menandakan panjang *word*.
5. RC5 harus dapat beroperasi dalam berbagai jumlah *round*. Jumlah *round* yang bervariasi memungkinkan pengguna untuk memanipulasi RC5 untuk menjadi lebih cepat dan aman.
6. RC5 harus dapat beroperasi dalam berbagai panjang kunci. Hal tersebut mengakibatkan panjang kunci *b* menjadi parameter dalam algoritma RC5.
7. RC5 haruslah berstruktur sederhana. Struktur yang sederhana tersebut belum tentu menghasilkan kemanan yang rendah. Namun, struktur sederhana akan memugkinkan analisis dan eveluasi yang cepat untuk menentukan kekuatan algoritma. [3].

8. RC5 harus hemat dalam penggunaan memori. Hal tersebut memungkinkan implemantasi RC5 kedalam smart-card atau perangkat lain yang memiliki keterbatasan memori.
9. RC5 harus mengimplementasikan metode *data-dependent rotations*. Metode RC5 merupakan kriptografi primitif yang merupakan sasaran pengkajian RC5. *Datadependent rotations* merupakan suatu teknik yang merotasi data secara sirkuler sebanyak N rotasi.

Ada 3 proses utama dalam RC5, yaitu perluasan kunci, enkripsi dan dekripsi. Perluasan kunci merupakan proses membangkitkan kunci internal dengan memanfaatkan komputasi rotasi *left regular shift* (<<<) dan *right regular shift* (>>>), dengan panjang kunci tergantung dari jumlah putaran. Kunci internal kemudian digunakan dalam proses enkripsi dan dekripsi. Proses enkripsi dibagi menjadi tiga, yaitu penjumlahan integer, XOR dan rotasi.

II.4.1. Proses Enkripsi

Untuk memahami cara kerja RC-5, dapat dimulai dengan melihat konsep dasar bagaimana RC-5 ini bekerja. RC-5 Menggunakan operasi dasar untuk proses enkripsi sebagai berikut :

1. Data yang akan dienkripsi dikembangkan menjadi 2 bagian bagian kiri dan bagian kanan dan dilakukan penjumlahan dengan *key word* yang yang telah diekspansi sebelumnya. Penjumlahan ditunjukan dengan tanda ``+``, dan disimpan di dua register A dan register B.
2. Kemudian dilakukan operasi EX-OR, EX-OR tersebut ditandai dengan tanda simbol ``⊕``.

3. Melakukan rotasi kekiri (*shift left*) sepanjang *y* terhadap *x word* yang ditandai dengan *x*<<< *y*. *y* merupakan interpretasi *modulo w* atau jumlah kata *w* dibagi 2. Dengan *lg[w]* ditentukan jumlah putaran yang dilakukan.
4. Tahap akhir dilakukan penggabungan untuk mendapatkan data yang telah dienkripsi.

Diasumsikan terdapat dua buah blok input sebesar *w* bit, *A* dan *B*. Dan diasumsikan juga bahwa pembentukan kunci internal telah dilakukan, sehingga array *S*[0...*t*-1] telah dihitung. Sehingga *pseudecode* untuk proses enkripsi seperti di bawah:

A ← A + KI[0]

B ← B + KI[1]

for i ←1 **to** r **do**

A←((A⊕B) <<< B) + KI[2i]

B←((B⊕A) <<< A) + KI[2i+1]

Endfor

II.4.2. Proses Deskripsi

Proses dekripsi dilakukan dengan konsep dasar sebagai berikut :

1. Data yang telah dienkripsi dikembangkan kembali menjadi 2 bagian dan disimpan di dua register A dan register B.
2. Kemudian dilakukan rotasi ke kanan sejumlah *r*.
3. Selanjutnya dilakukan operasi EX-OR, EX-OR tersebut yang ditandai dengan simbol ``⊕``.

4. Tahap akhir dilakukan pengurangan terhadap masing-masing register dengan key *word* yang ditunjukan dengan tanda ``-``, untuk mendapatkan *plaintext.* [4].

Algoritma pada proses dekripsi merupakan kebalikan dari proses enkripsi. Jika tadinya digeser ke kiri, maka pada proses dekripsi dilakukan pergeseran ke kanan (*right regular shift*).

for i ← r **downto** 1 **do**

B ← ((B – KI[2i+1])>>>A) ⊕ A

A←((A – KI[2i])>>>B) ⊕ B

endfor

B ←B - KI[1]

A ←A - KI[0]

Untuk mendekripsi cipherteks, diperlukan KI yang sama dengan KI saat mengenkripsi. Proses pembangkitan KI pada kedua proses tersebut juga sama.

II.4.3. Pembentukan Kunci Internal

K[0-1]...K[b] disalin ke tabel L[01]...L[b] dengan aturan di-*padding* dengan karakter 0 hingga ukuran L[i]menjadi w/2 bit. Sebagai contoh:

K[0] = k L[0] = k000

K[1] = r L[1] = r000

K[2] = i L[2] = i000

K[3] = p L[3] = p000

K[4] = t L[4] = t000

K[5] = o L[5] = o000

Kemudian, inisialisasi tabel kunci internal KI dengan ukuran t = 2r + 2 seperti berikut:

KI[0] ← P

for i ←1 **to** t – 1 **do**

KI[i] ← KI[i – 1]

Endfor

Algoritma pembentukan kunci interna menggunakan konstanta P dan Q yang didapatkan dari fungsi yang melibatkan bilangan irasional sebagai berikut:

$P = Odd[(e - 2)2^w]$

$Q = Odd[(f – 1)2^w]$

Keterangan:

e = 2.718281828459.....

F = 1.618033988749.....

Kemudian L dan S digabungkan dengan algoritma berikut:

i ← 0

j ← 0

X ← 0

Y ← 0

n ← 3*max(r,c)

for k ← 1 **to** n **do**

KI[i] ← (KI[i] + X + Y)<<< 3

X ← KI[i]

i ← (i + 1) **mod** t

L[j] ← (L[j] + X + Y)<<< 3

Y ← L[j]

J ← (j + 1) **mod** c

endfor

Keterangan:

max(r,c) adalah fungsi menentukan bilangan terbesar antara r dan c. c adalah nilai maksimal dari panjang kunci b dibagi 4.

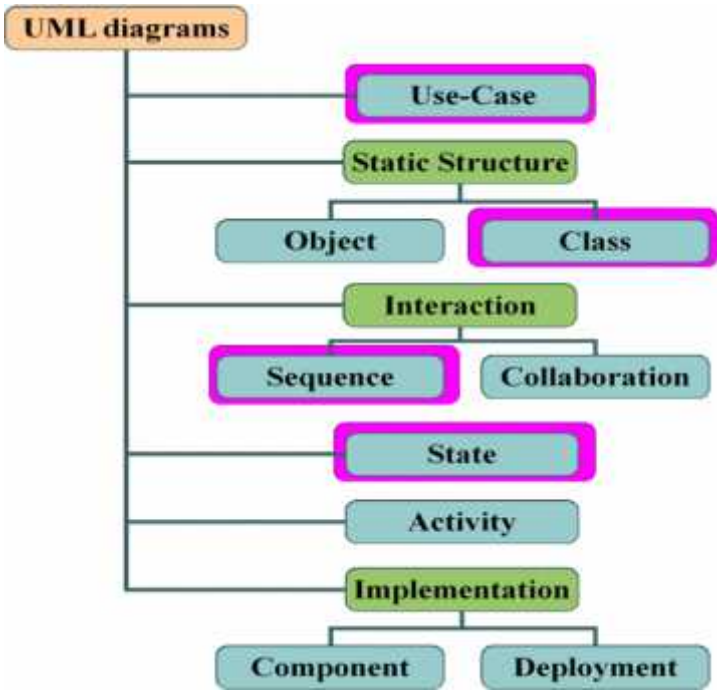
II.5. *Unified Modeling Languange* (UML)

Unified Modelling Language (UML) Menurut (Haviluddin) adalah suatu ala tuntuk memvisualisasikan dan mendokumentasikan hasil analisa dan desain yang berisi sintak dalam memodelkan sistem secara *visual* (Braun, *et. al*. 2012). Juga merupakan satu kumpulan konvensi pemodelan yang digunakanuntuk menentukan atau menggambarkan sebuah sistem *software* yang terkait dengan objek.

Sejarah UML sendiri terbagi dalam dua fase; sebelum dan sesudah munculnya UML. Dalam *fase* sebelum, UML sebenarnya sudah mulai diperkenalkan sejak tahun 1990an namun notasi yang dikembangkan oleh para ahli analisis dan desain berbeda-beda, sehingga dapat dikatakan belum memiliki standarisasi.

Saat ini sebagian besar para perancang sistem informasi dalam menggambarkan informasi dengan memanfaatkan UML diagram dengan tujuan utama untuk membantu tim proyek berkomunikasi, mengeksplorasi

potensi desain, dan memvalidasi desain arsitektur perangkat lunak atau pembuat program. Secara filosofi UML diilhami oleh konsep yang telah ada yaitu konsep permodelan *Object Oriented* karena konsep ini menganalogikan system seperti kehidupan nyata yang didominasi oleh obyek dan digambarkan atau dinotasikan dalam simbol-simbol yang cukup spesifik. Diagram Uml dapat dilihat pada gambar II.1 seperti dibawah in :



Gambar II.1. Diagram UML
(*Sumber : Haviluddin; 2013 : 2*)

II.5.1. Tujuan Pemanfaatan UML

Tujuan dari penggunaan diagram seperti diungkapan oleh Schmuller J. (2004), "*The purposeof the diagrams is to present multiple views of a system; this set of multiple views is called a model*". Berikut tujuan utama dalam desain UML adalah [5]

1. Menyediakan bagi pengguna (analisis dan desain sistem) suatu bahasa pemodelan *visual* yang ekspresif sehingga mereka dapat mengembangkan dan melakukan pertukaran model data yang bermakna.
2. Menyediakan mekanisme yang spesialisasi untuk memperluas konsep inti.
3. Karena merupakan bahasa pemodelan visual dalam proses pembangunannya maka UML bersifat independen terhadap bahasa pemrograman tertentu.
4. Memberika dasar formal untuk pemahaman bahasa pemodelan.
5. Mendorong pertumbuhan pasar terhadap penggunaan alat desain sistem yang berorientasi objek (OO).
6. Mendukung konsep pembangunan tingkat yang lebih tinggi seperti kolaborasi, kerangka, pola dan komponen terhadap suatu sistem.
7. Memiliki integrasi praktik terbaik.

II.5.2. Struktur Diagram

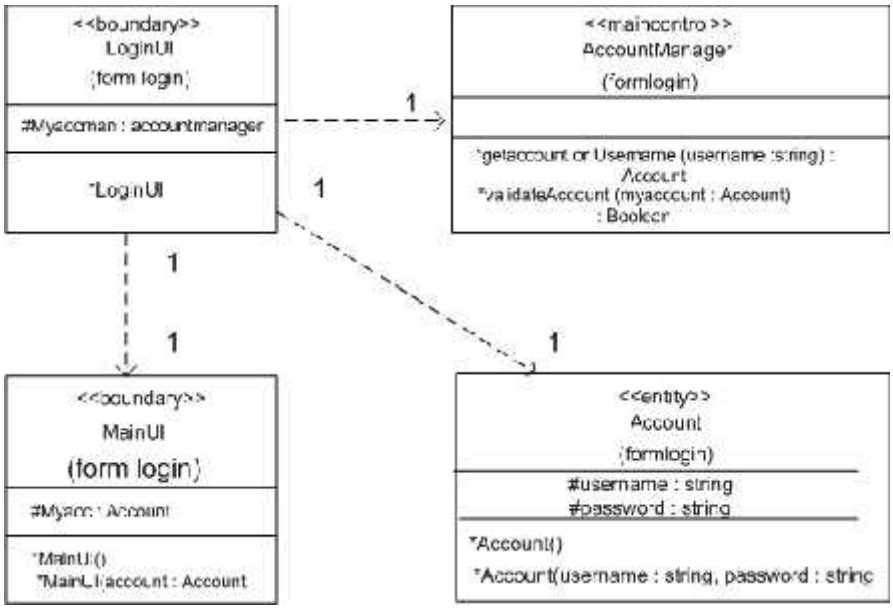
Menggambarkan elemen dari spesifikasi dimulai dengan kelas, obyek, dan hubungan mereka, dan beralih ke dokumen arsitektur logis dari suatu sistem. Struktur diagram dalam UML terdiri atas :

1. *Class Diagram*

   Class diagram menggambarkan struktur statis dari kelas dalam sistem anda dan menggambarkan atribut, operasi dan hubungan antara kelas.

Class diagram membantu dalam memvisualisasikan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. Selama tahap desain, *class diagram* berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat. *Class* memiliki tiga area pokok :

1. Nama (dan *stereo type*)
2. Atribut
3. Metoda

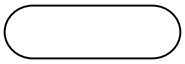
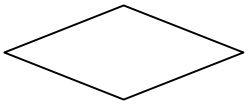


Gambar II.2. Notasi class diagram
(*Sumber : Haviluddin; 2013 : 3*)

II.6. *Activity* diagram

Menggambarkan aktifitas-aktifitas,objek, *state*, transisi *state* dan *event*. Dengan kata lain kegiatan diagram alur kerja menggambarkan perilaku sistem untuk aktivitas.

Tabel II.1. Activity Diagram

	*Activity*
	*Transition*
	*Decison*
	*Synchronization Bars*

(Sumber : Haviluddin; 2013 : 4)

UML memiliki seperangkat notasi yang akan digunakan ke dalam tiga kategori diatas yaitu struktur diagram, *behaviour* diagram dan *interaction* diagram. Berikut beberapa notasi dalam UML diantaranya :

1. *Actor*

Actor menentukan peran yang dimainkan oleh *user* atau sistem lain yang berinteraksi dengan subjek. *Actor* adalah segala sesuatu yang berinteraksi langsung dengan sistem aplikasi komputer, seperti orang, benda atau lainnya. Tugas *actor* adalah memberikan informasi kepada sistem dan dapat memerintahkan sistem untuk melakukan sesuatu tugas.

Gambar II.3. Notasi *actor*
(*Sumber : Haviluddin; 2013 : 6*)

2. *Class Diagram*

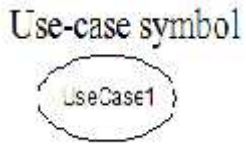
Notasi utama dan yang paling mendasar pada diagram UML adalah notasi untuk mempresentasikan suatu *class* beserta dengan atribut dan operasinya. *Class* adalah pembentuk utama dari sistem berorientasi objek

.

Gambar II.4. Notasi *class*
(*Sumber : Haviluddin; 2013 : 6*)

3. *Use Case* dan *Use Case Specification*

Use case adalah deskripsi fungsi dari sebuah si stem perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem disebut skenario. *Use case* merupakan awal yang sangat baik untuk setiap *fase* pengembangan berbasis objek, *design*, *testing*, dan dokumentasi yang menggambarkan kebutuhan sistem dari sudut pandang di luar sistem. Perlu diingat bahwa *use case* hanya menetapkan apa yang seharusnya dikerjakan oleh sistem, yaitu kebutuhan fungsional sistem dan tidak untuk menentukan kebutuhan non- fungsional, misalnya: sasaran kinerja, bahasa pemrograman dan lain sebagainya.



Gambar II.5. Notasi *use case*
(Sumber : Haviluddin; 2013 : 6)

1. *Realization*

Realization menunjukkan hubungan bahwa elemen yang ada di bagian tanpa panah akan merealisasikan apa yang dinyatakan oleh elemen yang ada di bagian dengan panah.

Gambar II.6. Notasi *realization*
(*Sumber : Haviluddin; 2013 : 6*)

2. *Interaction*

Interaction digunakan untuk menunjukkan baik aliran pesan atau informasi antar obyek maupun hubungan antar obyek.

Gambar II.7. Notasi *Interaction*
(*Sumber : Haviluddin; 2013 : 6*)

II.7. Bahasa Pemrograman Visual Basic 2010

Visual Basic merupakan salah satu bahasa pemrograman yang andal dan banyak digunakan oleh pengembang untuk membangun berbagai macam aplikasi

Windows. Visual Basic 2010 atau Visual Basic 9 adalah versi terbaru yang telah diluncurkan oleh Microsoft bersama C#, visual C++, dan Visual Web Developer dalam satu paket Visual Studio 2010 [6].

Visual Basic 2010 merupakan aplikasi pemrograman yang menggunakan teknologi *.NET Framework*. Teknologi *.NET Framework* merupakan komponen Windows yang terintegrasi serta mendukung pembuata, penggunaan aplikasi, dan halaman web. Teknologi *.NET Framework* mempunyai 2 komponen utama, yaitu CLR (*Common Language Runtime*) dan *Class Library*, CLR digunakan untuk menjalan aplikasi yang berbasis .NET, sedangkan *Library* adalah kelas pustaka atau perintah yang digunakan untuk membangun aplikasi.

Sebelum menginstall komputer harus memenuhi beberapa persyaratan agar Visual Basic 2010 dapat dijalankan dengan baik. Adapun, persyaratan *(System Requirements*) yang harus dipenuhi dapat Anda lihat pada Tabel II.2.

Tabel II.2. Sistem Requirements Visual Basic 2010

Sistem	Syarat Minimal	Syarat yang direkomendasikan
Arsitektur	X86 dan x64 (WOW)	
Sistem Operasi	Microsoft Windows XP Service Pack 2 Microsoft Windows Server 2003 Windows Vista	
Prosesor	CPU 1.6 GHz (Giga Hertz)	Windows XP dan Windows Server 2003:CPU 2,2 GHz atau yang lebih tinggi. Windows Vista : CPU 2,4 GHz
RAM	Windows XP dan Windows Server 2003 384 MB (Mega byte) Windows Vista : 768 MB	RAM 1024 MB / 1 GB atau yang lebih besar.
Harddisk	Tanpa MSDN Ruang Kosong harddisk pada drive	Kecepatan harddisk 7200 RPM atau yang

	penginstalan 2 GB. Sisa ruang harddisk kosong 1 GB Dengan MSDN Ruang kosong harddisk pada drive penginstalan 3,8 GB (MSDN diinstal full) 2,8 GB untuk menginstal MSDN default. Kecepatan Harddisk 5400 RPM.	lebih tinggi.
Display Layar	1024 x 768 display	1280 x 1024 display

(Sumber : Wahana Komputer ; 2012 : 2)

II.8. Keamanan Data

Keamanan data merupakan bagian dari perkembangan teknologi informasi. Ketika berpikir bahwa data yang dimiliki merupakan data yang sangat penting, semua berusaha untuk melindunginya agar jangan sampai jatuh ke tangan orang yang tidak bertanggung jawab. Tetapi buat sebagian orang, mereka justru tidak mengetahui sepenting apakah data yang mereka miliki. Karena ketidaktahuan tersebut, mereka baru menyadari bahwa data yang mereka miliki sangat penting setelah mengalami kecurian data dan mengalami kerugian. Data di sini bisa bersifat umum tidak terbatas pada data digital saja, tetapi juga seperti data diri (ktp, ijasah, sertifikat, dan lain-lain). Data yang menyangkut informasi pribadi tidak seharusnya diumbar sembarang seperti pada blog, situs jejaring pertemanan, email, selebaran, fotokopi KTP di buang sembarangan dan lain-lain. [7].

Masalah keamanan merupakan salah satu aspek terpenting dari sebuah sistem informasi. Masalah keamanan sering kurang mendapat 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. Apabile mengganggu performansi sistem, masalah keamanan sering tidak dipedulikan, bahkan ditiadakan.

Informasi menentukan hampir setiap elemen dari kehidupan manusia. Informasi sangat penting artinya bagi kehidupan karena tanpa informasi maka hampir semuanya tidak dapat dilakukan dengan baik. Contohnya, jika membeli tiket penerbangan dan membayarnya dengan menggunakan kartu kredit, informasi mengenai diri nantinya disimpan dan dikumpulkan serta digunakan oleh bank dan penerbangan. Demikian juga halnya saat membeli obat di apotik. Harus mendapat resep dari dokter dan memberikan resep tersebut ke pelayan apotik. Resep itu merupakan satu informasi yang disampaikan dokter ke pihak apotik tentang obat yang dibutuhkan.

Kemajuan sistem informasi memberikan banyak keuntungan bagi kehidupan manusia. Meski begitu, aspek negatifnya juga banyak, seperti kejahatan komputer yang mencakup pencurian, penipuan, pemerasan, kompetisi, dan banyak lainnya. Jatuhnya informasi ke pihak lain, misalnya lawan bisnis, dapat menimbulkan kerugian bagi pemilik informasi. Sebagai contoh, banyak informasi milik perusahaan yang hanya boleh diketahui oleh orang-orang tertentu di perusahaan tersebut, seperti misalnya informasi tentang produk yang sedang dalam pengembangan. Algoritma dan teknik yang digunakan untuk menghasilkan produk tersebut. Untuk itu keamanan dari sistem informasi yang digunakan harus terjamin dalam batas tertentu.

II.9. Data

Setiap pengguna komputer biasanya punya data pribadi yang disimpan di komputer atau di media penyimpanan mobile macam *flashdisk*. Data itu bisa terkait dengan kegiatan pribadi misalnya, imil, riwayat jelajah di internet, atau terkait dengan urusan kerja. Setiap orang tentulah punyak kriteria sendiri tentang data vital pribadinya.

Bagaimana cara menyimpan dan melindungi data pribadi itu pun mungkin berbeda-beda antara satu pengguna dengan lainnya. Namun, pada umumnya para pengguna menyimpan data pribadinya itu dengan cara menempatkannya dalam *folder* tersendiri di partisi tertentu pada harddisk atau di *flashdisk*. Bahkan tak sedikit pengguna yang melatakkan data pentingnya di *My Documents*.

Tentuk saja cara penempatan *file* penting macam itu bisa aman-aman saja sejauh komputer atau flashdisk yang ditempati data itu digunakan sendiri, dan tak mungkin hilang. Dalam keadaan tertentu terpaksa meminjamkan komputer atau *flashdisk* kepada orang lain. Pada kondisi macam ini, tentulah data penting berpotensi atau dilirik orang yang dipinjami. [3].