BAB II

TINJAUAN PUSTAKA

II.1. Perancangan

Perancangan adalah suatu proses pemilihan dan pemikiran yang menghubungkan fakta-fakta berdasarkan asumsi-asumsi yang berkaitan dengan masa datang dengan menggambarkan dan merumuskan kegiatan-kegiatan tertentu yang diyakini diperlukan untuk mencapai tujuan-tujuan tertentu dan menguraikan bagaimana pencapaiannya.(Anggraheni Rukmana ; 2012 : 2)

II.2. Sistem

Sistem adalah sebuah tatanan atau keterpaduan yang terdiri dari sejumlah komponen fungsional (dengan satuan fungsi atau tugas khusus) yang saling berhubungan dan secara bersama – sama bertujuan untuk memenuhi suatu proses atau pekerjaan tertentu. Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, artinya saling bekerja sama membentuk satu kesatuan. Suatu sistem mempunyai karakteristik atau sifat–sifat tertentu, yaitu mempunyai komponen–komponen (components), batas sistem (boundary), lingkungan luar sistem (environment), penghubung (interface), masukan (input), keluaran (output), pengolah (process) dan sasaran (objectives) atau tujuan (goal). Komponen–komponen sistem atau elemen–elemen sistem dapat berupa suatu sub sistem atau bagian–bagian dari sistem (Jogiyanto, 1999).(Hafsah ; 2011 : D-43)

II.3. Sistem Informasi

Sistem informasi adalah suatu kegiatan dari prosedur yang diorganisasikan bilamana dari eksekusi akan menyediakan informasi untuk mendukung pengambilan keputusan dan pengendalian dari dalam organisasi. (Anis nurhanafi; 2013 : 2)

Sebuah sistem informasi adalah sistem buatan manusia yang berisi himpunan terintegrasi dari komponenkomponen manual dan komponen-komponen terkomputerisasi yang bertujuan untuk mengumpulkan data, memproses data, dan menghasilkan informasi untuk pemakai. (Atik Rusmayanti ; 2012 : 2)

II.4. Kehadiran Dosen

Kehadiran bagi dosen dalam belajar mahasiswa akan memberikan pengaruh terhadap persentase nilai ≥ B apabila kehadiran dosen dalam memberikan kuliah di atas 12.498403 (0.7812 x 16) atau di atas 12 kali kehadiran. Sebaliknya, jika kehadiran dosen dalam memberikan kuliah kurang dari 12 kali, maka kemauanbelajar mahasiswa tidak akan memberikan pengaruh terhadap peningkatan pencapaian persentase nilai ≥ B. (Atik Rusmayanti ; 2012 : 89)

II.5. Konsep Dasar Keamanan Data

Security dikaitkan dengan pengamanan data, sementara *intelligence* dikaitkan dengan pencarian (pencurian, penyadapan) data. Keduanya sama pentingnya. Bagi sebuah perusahaan, biasanya masalah pengamanan data yang lebih dipentingkan. Sementara bagi militer dan intel, masalah penyadapan data

merupakan hal yang penting juga karena ini menyangkut keamanan negara. Hal ini menimbulkan masalah baru seperti masalah privasi dan keamanan negara, masalah *spy versus spy*. Lubang keamanan *Security Hole* dapat terjadi karena beberapa hal yaitu : Salah Desain, Implementasi kurang baik, Salah konfigurasi, Salah menggunakan program atau system. ada 2 cara pengamanan data yaitu :

1. Steganography

Pengamanan dengan menggunakan *steganografi* membuat seolah-oleh pesan rahasia tidak ada atau tidak nampak. Padahal pesan tersebut ada. Hanya saja kita tidak sadar bahwa ada pesan tersebut di sana. Pesan rahasia dapat juga dikirimkan dengan mengirim surat pembaca ke sebuah surat kabar. Huruf awal setiap kalimat (atau bisa juga setiap kata) membentuk pesan yang ingin diberikan. Cara lain adalah dengan membuat puisi dimana huruf awal dari setiap baris membentuk kata-kata pesan sesungguhnya. Hal yang sama dapat dilakukan dengan membuat urutan gambar buah dimana pesan tersebut merupakan gabungan dari huruf awal dari nama buah tersebut.

Di dunia digital, steganografi muncul dalam bentuk *digital watermark*, yaitu tanda digital yang disisipkan dalam gambar (*digital image*) atau suara. Hak cipta (*copyright*) dari gambar dapat disisipkan dengan menggunakan high-bit dari pixel yang membentuk gambar tersebut. Gambar terlihat tidak berbeda -karena kemampuan (atau lebih tepatnyaketidakmampuan) mata manusia yang tidak dapat membedakan satu bit saja -akan tetapi sebenarnya mengandung pesan-pesan tertentu. *Steganografi* juga muncul dalam aplikasi digital audio, seperti misalnya untuk melindungi lagu dari pembajakan. Contoh lain adalah menyisipkan

informasi sudah berapa kali lagu tersebut didengarkan. Setelah sekian kali didengarkan, maka pengguna harus membayar sewa lagu. (Meskipun pendekatan ini masih bermasalah.)

2. Cryptography

Kriptografi (*cryptography*) merupakan ilmu dan seni untuk menjaga pesan agar aman. (*Cryptography is the art and science of keeping messages secure*) "*Crypto*" berarti "*secret*" (rahasia) dan "*graphy*" berarti "*writing*" (tulisan). Para pelaku atau praktisi kriptografi dilakukan dengan dua cara, yaitu transposisi dan subsitusi. Pada penggunaan transposisi, posisi dari huruf yang diubah-ubah, sementara pada substitusi, huruf (atau kata) digantikan dengan huruf atau simbol lain. Jadi bedanya dengan steganografi adalah pada kriptografi pesan nampak. Hanya bentuknya yang sulit dikenali karena seperti diacak-acak.

II.6. Kecurangan /Pencurian dan Konsep Serangan Keamanan

1. Kecurangan dan Pencurian antara lain.
   a. *Physical Access Attacks*

      serangan yang mencoba mendapatkan akses melalui jalur fisik, contoh : *Wiretapping*, yaitu : usaha untuk dapat mengakses data melalui media transmisi (kabel). *Server hacking*, usaha untuk mengakses *resource server* langsung secara fisik. *Vandalism*, serangan dengan tujuan rusaknya sistem secara fisik.

   b. *Dialog Attacks*

      serangan yang dilakukan saat pihak pengirim dan penerima men-transmisi-kan datanya, contoh : *Eavesdropping*, yaitu menguping dan mengintip data

yang sedang ditransmisikan. *Impersonation,* yaitu serangan dengan cara berpura-pura (menipu) mengaku sebagai orang lain. *Message Alteration,* yaitu serangan dengan cara mengubah data yang dikirim pihak lain sebelum sampai ke tujuannya.

c. *Penetration Attacks*

serangan yang mencoba menembus pertahanan system melalui mekanisme jaringan, contoh : *Scanning(Probing) ,* yaitu serangan dengan cara pelacakan kemungkinan mendapatkan celah kelemahan suatu sistem yang akan diserang. *of Service (DoS),* yaitu serangan dengan tujuan men-*down*-kan server dengan cara meminta /me-request service terus-menerus. *Brute Force,* usaha penembusan dengan cara coba-coba, misalnya mencoba semua kemungkinan password.

d. *Maliciuos Code Attacks*

serangan yang dilakukan melalui suatu kode program yang diselipkan ke program lain, contoh : kode program yang menumpang ke program lain, yang dapat memperbanyak dirinya sendiri ke program lain, dan melakukan hal yang tidak diinginkan. *Worm,* yaitu program yang dapat meng-copy dirinya sendiri dan mengirimkannya dari satu komputer ke komputer lain melalui jaringan.

e. Social Enggineering

serangan yang dilakukan melalui aktivitas sosial, contoh : *Password Theft,* yaitu mencuri password seseorang yang memiliki account. *Information Theft,* yaitu mencuri informasi dari seseorang/orang dalam.

2. Konsep Serangan Keamanan antara lain.

Serangan terhadap keamanan, menurut W. Stalling, yaitu :

a. Interruption

   Perangkat sistem menjadi rusak atau tidak tersedia. Serangan ditujukan kepada ketersediaan (availability) dari sistem.

   Contoh serangan adalah “denial of service attack

b. Interception

   Pihak yang tidak berwenang berhasil mengakses aset atau informasi.

   Contoh dari serangan ini adalah penyadapan (wiretapping).

c. Modification

   Pihak yang tidak berwenang tidak saja berhasil mengakses, akan tetapi dapat juga mengubah (tamper) aset.

   Contoh dari serangan ini antara lain adalah mengubah isi dari web site dengan pesan - pesan yang merugikan pemilik web site.

d. Fabrication

   Pihak yang tidak berwenang menyisipkan objek palsu ke dalam sistem.

   Contoh dari serangan jenis ini adalah memasukkan pesan pesan palsu seperti e-mail palsu ke dalam jaringan komputer.

II.7. Algoritma Base64

Transformasi *Base64* merupakan salah satu algoritma untuk *Encoding* dan *Decoding* suatu data ke dalam format ASCII, yang didasarkan pada bilangan dasar 64 atau bisa dikatakan sebagai salah satu metoda yang digunakan untuk melakukan encoding (penyandian) terhadap data binary. Karakter yang dihasilkan

pada transformasi *Base64* ini terdiri dari A..Z, a..z dan 0..9, serta ditambah dengan dua karakter terakhir yang bersimbol yaitu + dan / serta satu buah karakter sama dengan (=) yang digunakan untuk penyesuaian dan menggenapkan data binary atau istilahnya disebut sebagai pengisi pad.

Karakter simbol yang akan dihasilkan akan tergantung dari proses algoritma yang berjalan. Kriptografi Transformasi *Base64* banyak digunakan di dunia internet sebagai media data format untuk mengirimkan data, ini dikarenakan hasil dari *Base64* berupa *Plaintext*, maka data ini akan jauh lebih mudah dikirim, dibandingkan dengan format data yang berupa bineri. Dalam Implementasinya beberapa contoh dalam Transformasi *Base64*, yang antara lain adalah sebagai berikut(Wahana Komputer, 2010).

1. PEM (*Privacy-Enhaced Mail*) adalah protocol pertama dengan teknik Base64 yang didasarkan pada RFC 989, yang terdiri dari 7 karakter (7- bit) yang digunakan pada SMTP dalam transfer data tapi untuk sekarang PEM sudah tidak menggunakan RFC 989 tapi sudah di ganti dengan RFC 1421 yang menggukana karakter A_Z, a-z, 0..9.
2. MIME (*Multi Purpose Mail Extension*) didasarkan pada RFC 2045. Teknik encoding Base64 MIME, mempunyai konsep yang berdasarkan RFC 1421 versi PEM. Sedangkan MIME diakhiri dengan Padding "=" pada hasil akhir encodingnya.
3. UTF-7 didasarkan pada RFC 2152, yang umumnya disebut "MODIFICATION BASE" UTF-7 menggunakan karakter MIME, tidak memakai padding"=", karakter "=" digunakan sebagia escape untuk encoding.

4. OpenPGP (*PGP Prety Good Privacy*) dirancang pada RFC 2440, yang menggukan Coding 64 Radix atau ASCI Amor. Teknik encodingnya didasarakan pada MIME tetapi ditambah dengan 24 bit CRC Cheksum.

Nilai Cheksum dihitung dari data Input sebelum dilakukan Proses Encoding. Dalam *Encoding Base64* dapat dikelompokkan dan dibedakan menjadi beberapa kriteria yang tertera dan dapat dilihat di dalam Tabel II.1

Tabel.II.1. *Encoding Base64* (Josefsson, 2003)

Data 6 bit	Karakter Encoding 64	Data 6 bit	Karakter Encoding 64
0	A	33	h
1	B	34	i
2	C	35	j
3	D	36	k
4	E	37	l
5	F	38	m
6	G	39	n
7	H	40	o
8	I	41	p
9	J	42	q
10	K	43	r
11	L	44	s
12	M	45	t
13	N	46	u
14	O	47	v
15	P	48	w
16	Q	49	x
17	R	50	y
18	S	51	z
19	T	52	0
20	U	53	1
21	V	54	2
22	W	55	3
23	X	56	4
24	Y	57	5
25	Z	58	6
26	A	59	7
27	B	60	8
28	C	61	9
29	D	62	+
30	E	63	/
31	F	(pad)	=
32	G		

Sumber : (Febrian Wahyu ; 2012 : 49)

Teknik *encoding Base64* sebenarnya sederhana, jika ada satu (*string*) *bytes* yang akan disandikan ke *Base64* maka caranya adalah(Ariyus, 2008).

1. Pecah *string bytes* tersebut ke per-3 *bytes*.
2. Gabungkan 3 *bytes* menjadi 24 *bit*. Dengan catatan 1 *bytes* = 8 bit, sehingga 3 x 8 = 24 *bit*.
3. Lalu 24 *bit* yang disimpan di-*buffer* (disatukan) dipecah-pecah menjadi 6 *bit*, maka akan menghasilkan 4 pecahan.
4. Masing masing pecahan diubah ke dalam nilai *decimal*, imana maksimal nilai 6 *bit* dalah 63.
5. Terakhir, jadikan nilai nilai desimal tersebut menjadi indeks untuk memilih karakter penyusun dari *base64* dan maksimal adalah 63.

Dan seterusnya sampai akhir *string bytes* yang mau kita konversikan. Jika ternyata dalam proses *encoding* terdapat sisa pembagi, maka tambahkan sebagai penggenap sisa tersebut karakter =. Maka terkadang pada *base64* akan muncul satu atau dua karakter = (). *(Febrian Wahyu ; 2012 : 49)*

II.7.1. *Java*

Java memiliki cara kerja yang unik dibandingkan dengan bahasa perograman lainya yaitu bahasa perograman *java* bekerja mengunakan *interptreter* dan juga *compiler* dalam proses pembuatan program, *Interpreter java* dikenal sebagai perograman *bytecode* yaitu dengan cara kerja mengubah paket *class* pada *java* dengan *extensi*. *Java* menjadi *.class*, hal ini dikenal sebagai *class bytecode*, yaitunya *class* yang dihasilkan agar program dapat dijalankan pada semua jenis perangkat dan juga *platform*, sehingga program *java* cukup ditulis sekali namun mampu bekerja pada jenis lingkungan yang berbeda. (*Defni, Indri Rahmayun* ; 2014 : 64)

II.8. Keamanan Data dan Jaringan Komputer

Keamanan dan kerahasiaan data merupakan salah satu aspek terpenting dalam bidang komunikasi, khususnya komunikasi yang menggunakan media komputer. Salah satu bidang ilmu pengetahuan yang digunakan untuk mengamankan data adalah kriptografi. Kriptografi merupakan ilmu pengetahuan yang menggunakan persamaan matematis untuk melakukan proses enkripsi dandekripsi data. Enkripsi merupakan proses untuk mengubah plainteks (data yang dapat dibaca) menjadi cipherteks (data yang tidak bisa dibaca) dan dekripsi merupakan kebalikan dari enkripsi, yaitu merubah cipherteks kembali menjadi plainteks. (Busran ; 2012 : 32)

Data merupakan deskripsi dari sesuatu kejadian yang kita hadapi, sementara data bisnis didefinisikan sebagai deksripsi organisasi tentang suatu (resources) dan kejadian (transactions) yang terjadi, Data merupakan bentuk yang masih mentah yang belum dapat bercerita banyak, sehingga perlu diolah lebiih lanjut. Data diolah melalui suatu model untuk menghasilkan informasi. (Anastasia Lipursari ; 2013 : 28)

Jaringan komputer merupakan komunikasi data antar komputer, yaitu minimal 2 komputer. Jaringan komputer dapat dilakukan melalui media kabel ataupun nirkabel (wireless). Pada sistem antrian rumah sakit ini, jaringan komputer dilakukan melalui media kabel antara Komputer LAN. Disebut *Metropolitan Area Network* karena Jenis Jaringan Komputer MAN ini biasa digunakan untuk menghubungkan jaringan komputer dari suatu kota ke kota lainnya. Untuk dapat membuat suatu jaringan MAN, biasanya diperlukan adanya

operator telekomunikasi untuk menghubungkan antar jaringan komputer. (Mardison, S. Kom ; 2012 : 58-59)

II.8.1. Jaringan Komputer Berdasarkan Skala

1. Jaringan LAN

LAN (*Local Area Network*) adalah suatu kumpulan komputer, dimana terdapat beberapa unit komputer (*client*) dan 1 unit komputer untuk bank data (*server*). Antara masing-masing *client* maupun antara *client* dan *server* dapat saling bertukar *file* maupun saling menggunakan printer yang terhubung pada unit-unit komputer yang terhubung pada jaringan LAN.

2. Jaringan MAN

Metropolitan Area Network atau MAN, merupakan Jenis Jaringan Komputer yang lebih luas dan lebih canggih dari Jenis Jaringan Komputer LAN. Disebut *Metropolitan Area Network* karena Jenis Jaringan Komputer MAN ini biasa digunakan untuk menghubungkan jaringan komputer dari suatu kota ke kota lainnya. Untuk dapat membuat suatu jaringan MAN, biasanya diperlukan adanya operator telekomunikasi untuk menghubungkan antar jaringan komputer.

3. Jaringan WAN

WAN (*Wide Area Network*) adalah kumpulan dari LAN dan/atau *Workgroup* yang dihubungkan dengan menggunakan alat komunikasi modem dan jaringan Internet, dari/ke kantor pusat dan kantor cabang, maupun antar kantor cabang. Dengan sistem jaringan ini, pertukaran data antar kantor dapat dilakukan dengan cepat serta dengan biaya yang *relative* murah. Sistem jaringan ini dapat menggunakan jaringan *Internet* yang sudah ada, untuk menghubungkan antara

kantor pusat dan kantor cabang atau dengan PC Stand *Alone/Notebook* yang berada di lain kota ataupun negara. (*Mardison, al husni ; 2012 : 59*)

II.8.2. Jaringan Komputer Bertopologi / Pemasangan

1. Bus

Topologi bus merupakan topologi yang banyak dipergunakan pada masa penggunaan kabel sepaksi menjamur. Dengan menggunakan T-*Connector* (dengan terminator 50ohm pada ujung *network*), maka komputer atau perangkat jaringan lainnya bisa dengan mudah dihubungkan satu sama lain. Instalasi jaringan Bus sangat sederhana, murah dan maksimal terdiri atas 5-7 komputer. Kesulitan yang sering dihadapi adalah kemungkinan terjadinya tabrakan data karena mekanisme jaringan relative sederhana dan jika salah satu node putus maka akan mengganggu kinerja dan trafik seluruh jaringan.

2. *Ring*

Topologi cincin adalah topologi jaringan berbentuk rangkaian titik yang masing-masing terhubung ke dua titik lainnya, sedemikian sehingga membentuk jalur melingkar membentuk cincin. Pada topologi cincin, komunikasi data dapat terganggu jika satu titik mengalami gangguan.

3. *Star*

Topologi bintang merupakan bentuk topologi jaringan yang berupa *konvergensi* dari *node* tengah ke setiap *node* atau pengguna. Topologi jaringan bintang termasuk topologi jaringan dengan biaya menengah. (Mardison, Al Husni ; 2012 : 59)

4. *Mesh*

Topologi jala atau Topologi mesh adalah suatu bentuk hubungan antar perangkat dimana setiap perangkat terhubung secara langsung ke perangkat lainnya yang ada di dalam jaringan. Akibatnya, dalam topologi *mesh* setiap perangkat dapat berkomunikasi langsung dengan perangkat yang dituju (*dedicated links*).

5. Pohon

Topologi Pohon adalah kombinasi karakteristik antara topologi bintang dan topologi bus. Topologi ini terdiri atas kumpulan topologi bintang yang dihubungkan dalam satu topologi bus sebagai jalur tulang punggung atau *backbone*. Komputer-komputer dihubungkan ke *hub*, sedangkan *hub* lain di hubungkan sebagai jalur tulang punggung. Topologi jaringan ini disebut juga sebagai topologi jaringan bertingkat. Topologi ini biasanya digunakan untuk interkoneksi antar sentral dengan hirarki yang berbeda. Untuk hirarki yang lebih rendah digambarkan pada lokasi yang rendah dan semakin keatas mempunyai hirarki semakin tinggi. Topologi jaringan jenis ini cocok digunakan pada sistem jaringan komputer.

6. *Linier*

Jaringan komputer dengan topologi runtut (*linear topology*) biasa disebut dengan topologi bus beruntut, tata letak ini termasuk tata letak umum. Satu kabel utama menghubungkan tiap titik sambungan (komputer) yang dihubungkan dengan penyambung yang disebut dengan Penyambung-T dan pada ujungnya harus diakhiri dengan sebuah penamat (*terminator*). Penyambung yang digunakan

berjenis BNC (*British Naval Connector*: Penyambung *Bahari Britania*), sebenarnya BNC adalah nama penyambung bukan nama kabelnya, kabel yang digunakan adalah RG 58 (Kabel *Sepaksi Thinnet*). Pemasangan dari topologi bus beruntut ini sangat sederhana dan murah tetapi sebanyaknya hanya dapat terdiri dari 5-7 komputer. (*Mardison, Al Husni ; 2012 : 59*)

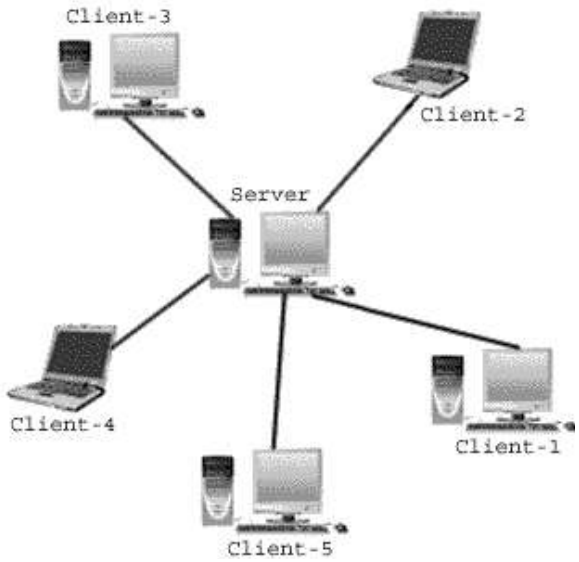
II.9. *Client Server*

Client – Server adalah bentuk *distributed computing* dimana sebuah program (*client*) berkomunikasi dengan program lain (*server*) dengan tujuan untuk bertukar informasi, pada umumnya sebuah *client* memiliki tugas sebagai berikut :

1. Menterjemahkan permintaan pengguna ke dalam bentuk *protocol* yang sesuai.
2. Mengirimkan permintaan pengguna ke *server*.
3. Menunggu respon dari *server*.

Kata *client* juga sering disebut dengan kata *host* yang menandakan bahwa *device* tersebut tersambung dalam sebuah jaringan. Sedangkan sebuah *server* memiliki tanggung jawab sebagai berikut :

1. Mendengarkan permintaan dari *client*.
2. Memproses permintaan tersebut.
3. Mengembalikan hasil proses tersebut ke client. (Painem ; 2013 : 16-17)



Gambar II.1. Model *Client Server*
Sumber : (Painem ; 2013 : 17)

II.10. UML *(Unified Modelling Language)*

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C. Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan

bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*). Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: *metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock,* dsb. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan group/perusahaan lain yang menggunakan metodologi yang berlainan. Dimulai pada bulan Oktober 1994 *Booch, Rumbaugh dan Jacobson*, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease *draft* pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh *Object Management Group* (OMG – http://www.omg.org). Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. *Booch, Rumbaugh* dan *Jacobson* menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma

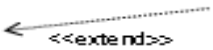
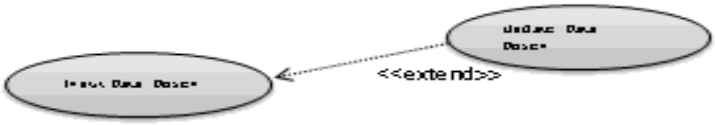
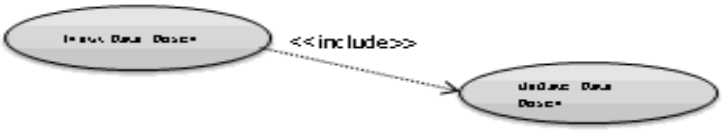
menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek.(Yuni Sugiarti ; 2013 : 33)

Dalam pembuatan skripsi ini penulis menggunakan diagram *Use Case* yang terdapat di dalam UML. Adapun maksud dari *Use Case* Diagram diterangkan dibawah ini.

1. *Use Case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya *login* ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem. Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri.

Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain. *(Yuni Sugiarti ; 2013 : 41)*

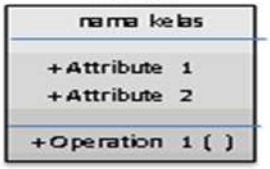
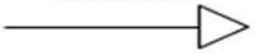
Simbol	Deskripsi
Use Case 	fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit dan aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama use case
Aktor  nama aktor	orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat 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 walaupun 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 menunjukan pada use case yang dituju contoh : 
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, contoh : 

Gambar II.2. *Use Case* Diagram
Sumber : (Yuni Sugiarti ; 2013 ; 42)

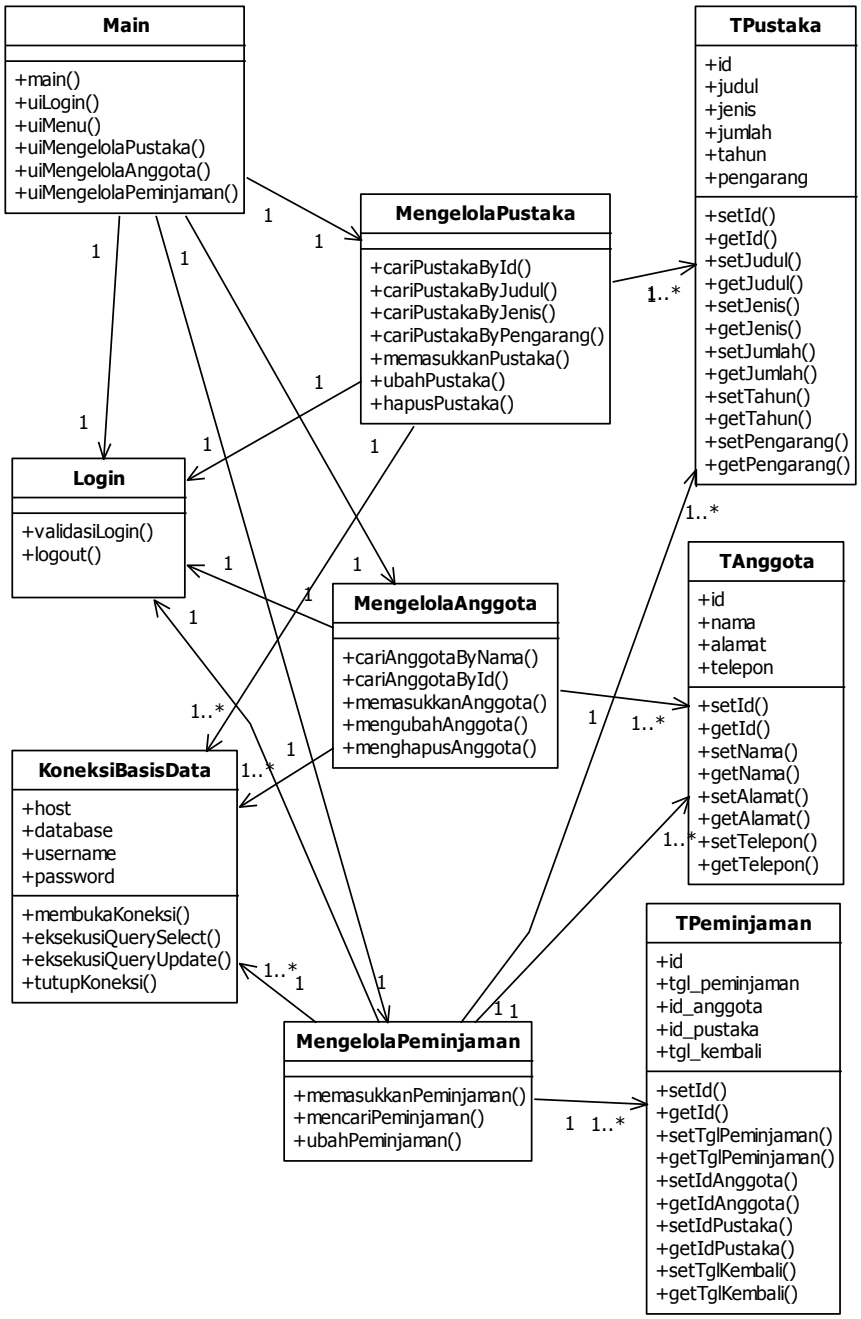
2. *Class Diagram*

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. Berikut adalah simbol-simbol pada diagram kelas :

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 	relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan multiplicity
Asosiasi berarah/directed asosiasi	relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai 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)

Gambar II.3. *Class* Diagram
Sumber : (Yuni Sugiarti ; 2013 : 59)



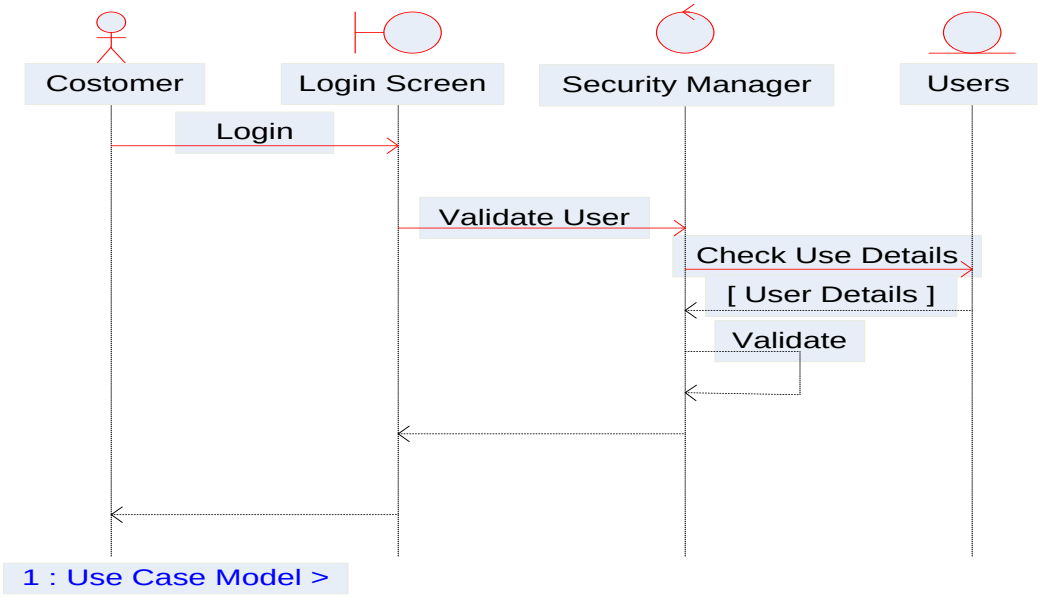
Gambar II.4. Contoh Class Diagram
Sumber : (Yuni Sugiarti ; 2013 : 63)

3. *Sequence Diagram*

Diagram *Sequence* menggambarkan kelakuan/prilaku 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 diinstansiasi menjadi objek itu.

Banyaknya diagram *sequence* yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram *sequence* sehingga semakin banyak *use case* yang didefinisikan maka diagram *sequence* yang harus dibuat juga semakin banyak.



Gambar II.5. Contoh *Sequence* Diagram
Sumber : (Yuni Sugiarti ; 2013 : 63)

4. *Activity Diagram*

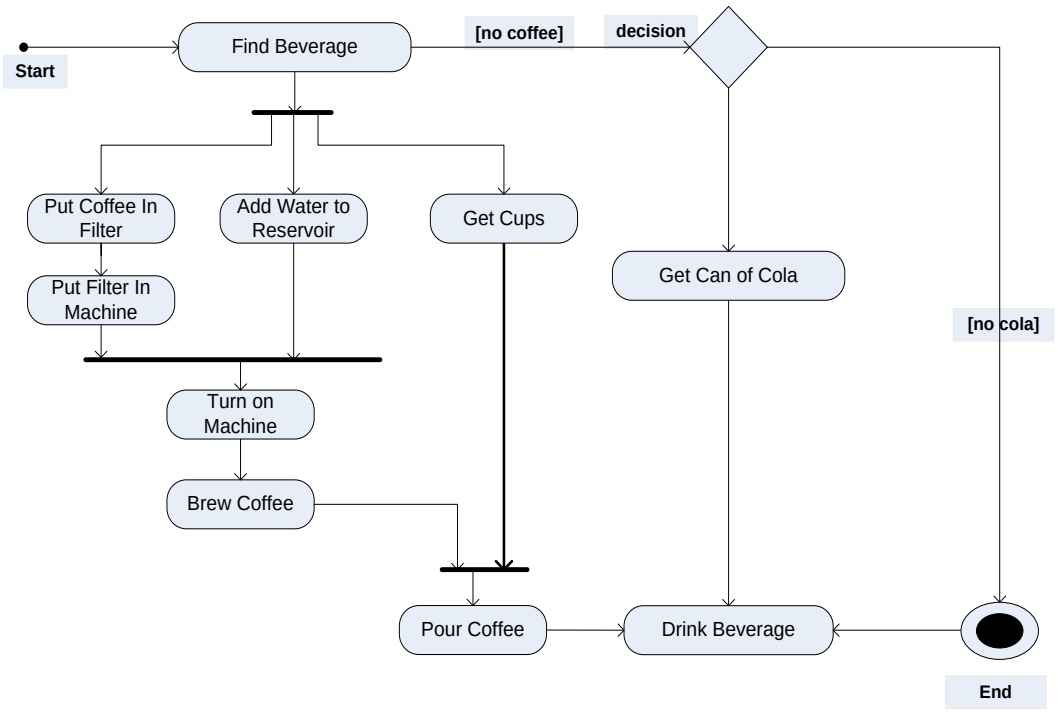
Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

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.

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.

Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal.

Activity diagram dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.



Gambar II.6. Activity Diagram
Sumber : (Yuni Sugiarti ; 2013 : 76)