BAB II

TINJAUAN PUSTAKA

II.1. Kompresi Data

Kompresi merupakan suatu upaya untuk mengurangi jumlah *bit* yang digunakan untuk menyimpan atau mentransmisikan data. Kompresi data meliputi berbagai teknik kompresi yang diterapkan dalam bentuk perangkat lunak (*software*) maupun perangkat keras (*hardware*). Bila ditinjau dari sisi penggunaannya, kompresi data bisa bersifat umum untuk segala keperluan atau bersifat khusus untuk keperluan tertentu. Keuntungan data yang terkompresi antara lain mengurangi *bottleneck* pada proses I/O dan transmisi data, penyimpanan data lebih hemat ruang, mempersulit pembacaan data oleh pihak yang tidak berkepentingan, dan memudahkan distribusi data dengan media *removable* seperti *flashdisk*, CD, DVD, dan lain-lain. *Kandaga, Tjatur* (2006 : 81)

File adalah kumpulan *record* sejenis yang mempunyai panjang elemen dan atribut yang sama, namun *value*-nya berbeda. *Kusrini* (2007:13)

File adalah data-data yang tersimpan dalam media yang mempunyai informasi besar *file*, tanggal dan jam penyimpanan *file*, nama *file*, ciri *file* (ciri aplikasi yang membuat), dan attribut *file*. *Hendrayudi* (2008 : 3)

Pengiriman data hasil kompresi dapat dilakukan jika pihak pengirim atau yang melakukan kompresi dan pihak penerima memiliki aturan yang sama dalam hal kompresi data. Pihak pengirim harus menggunakan algoritma kompresi data

yang sudah baku, data yang sama dengan pengirim sehingga data yang diterima dapat dibaca kembali dengan benar. *Anton* (2005)

Misalnya terdapat kata “Hari ini adalah hari Jum’at. Hari Jum’at adalah hari yang menyenangkan”. Jika ditelaah lagi, kalimat tersebut memiliki pengulangan karakter seperti karakter pembentuk kata hari, hari jumat, dan adalah. Dalam teknik sederhana kompresi pada perangkat lunak, kalimat di atas dapat diubah menjadi pola sebagai berikut “# ini $ %. % $ # ya@ menyena@kan”. Di mana dalam kalimat diatas, karakter pembentuk hari diubah menjadi karakter #, hari Jum’at menjadi %, adalah menjadi $, ng menjadi @. Saat berkas ini akan dibaca kembali, maka perangkat lunak akan mengembalikan karakter tersebut menjadi karakter awal dalam kalimat. *Jando* (2005).

II.2. Jenis Teknik Kompresi

Menurut *Restyandito* (2008), Teknik kompresi secara umum dapat diklasifikasikan menjadi tiga bagian yaitu:

a. *Entropy coding*, adalah teknik kompresi yang menggunakan proses *lossless*. Tekniknya tidak berdasarkan pada media dengan spesifikasi dan karakteristik tertentu namun berdasarkan urutan data serta tidak memperhatikan semantik data.
b. *Source coding*, adalah teknik kompresi dengan menggunakan proses *lossy*. Teknik ini berkaitan dengan data semantik (arti data) dan media.

c. *Hybrid coding* adalah teknik kompresi dengan menggunakan kombinasi atau gabungan dari *entropy coding* dan *source coding*.

Berdasarkan *output*nya, teknik kompresi dapat dibedakan menjadi dua jenis yaitu :

a. Teknik kompresi *Lossy*

Kompresi menggunakan *lossy,* beberapa bagian data asli hilang ketika berkas di *decoded*. Keuntungan dari algoritma ini adalah bahwa rasio kompresi cukup tinggi. Hal ini dikarenakan cara algoritma *lossy* yang mengeliminasikan beberapa data dari suatu berkas. Namun data yang dieliminasikan biasanya adalah data yang kurang diperhatikan atau di luar jangkauan manusia, sehingga pengeliminasian data tersebut kemungkinan besar tidak akan mempengaruhi manusia yang berinteraksi dengan berkas tersebut. *Linawati* (Vol. 9 : No. 1 : 2005)

b. Teknik kompresi *Lossless*

Kompresi menggunakan *lossless* menjamin bahwa berkas yang dikompresi dapat selalu dikembalikan ke bentuk aslinya. Algoritma *lossless* digunakan untuk kompresi berkas *text*, seperti program komputer (berkas *zip*, *rar*, dll), karena jika ingin mengembalikan berkas yang telah dikompres ke status aslinya. Kadangkala ada data-data yang setelah dikompresi dengan teknik ini ukurannya menjadi lebih besar atau sama. Berdasarkan teknik pengkodean/pengubahan simbol yang digunakan.

II.3. Metode Kompresi Algoritma LZW

Teknik kompresi LZW merupakan turunan dari teknik LZ78. Pada LZW digunakan kamus yang terdiri dari frase-frase yang terus-menerus dimasukkan kedalam kamus selama proses pengkodean/pendekodean. Frase baru akan terus-menerus dimasukkan kedalam kamus sampai tempat yang dicadangkan untuk kamus habis terpakai. Kamus pada LZW sudah diisi dengan frase-frase yang terdiri dari sebuah karakter yaitu seluruh karakter ASCII yang ada sebelum proses pengkodean/pendekodean dimulai. Frase-frase tersebut memiliki kode sesuai dengan kode ASCII dari karakter-karaktemya. Dengan cara ini maka tidak ada satu karakterpun yang tidak dapat langsung dikodekan oleh pengkode LZW. Kamus berupa sebuah pohon cabang-banyak (*multiway tree*), dan digunakan *hash function* dalam mencari simpul anak. *Kandaga, Tjatur* (2006 : 85)

Struktur yang digunakan dalam proses pengkodean/pendekodean algoritma LZW adalah sbb:

```
BEGIN
        S = next input character;
        While not EOF
        {
        C = next input character;
        If s + c exists in the diactionary
              S = s + c
                 Else
                 {
                      Output the code for s;
                      Add string s + c to the dictionary with a new code
                      S = c;
                 }
        }
END
```

Pendekatan ini bersifat adaptif dan efektif karena banyak karakter dapat dikodekan dengan mengacu pada *string* yang telah muncul sebelumnya dalam teks. Prinsip kompesi tercapai jika referensi dalam bentuk *pointer* dapat disimpan dalam jumlah *bit* yang lebih dibandingkan *string* aslinya. Sebagai contoh, *string* "ABBABABAC" akan dikompresi dengan LZW. Isi *dictionary* pada awal proses diset dengan tiga karakter dasar yang ada: "A", "B", "C". Tahapan proses kompresi ditunjukkan pada tabel II.1: *Linawati* (Vol. 9 : No. 1 : 2005)

Tabel II.1 : Tahapan Kompresi LZW

Langkah	**Posisi**	**Karakter**	***Dictionary***	***Output***
1	1	A	[4] A B	[1]
2	2	B	[5] B B	[2]
3	3	B	[6] B A	[2]
4	4	A	[7] A B A	[4]
5	6	A	[8] A B A C	[7]
6	9	C	----	[3]

Sumber : Linawati (Vol. 9 : No. 1 : 2005)

Kolom posisi menyatakan posisi sekarang dari *stream* karakter dan kolom karakter menyatakan karakter yang terdapat pada posisi tersebut. Kolom *dictionary* menyatakan string baru yang sudah ditambahkan ke dalam *dictionary* dan nomor indeks untuk string tersebut ditulis dalam kurung siku. Kolom *output* menyatakan kode *output* yang dihasilkan oleh langkah kompresi.

```
BEGIN
        S = NULL;
        while not EOF{
                K = NEXT INPUT CODE;
                Entry = dictionary entry for K;
                Ouput entry;
                if(s != NULL)
                        add string s + entry[0] to dictionary with new code
                        S = Entry;
        }
END
```

II.4. Metode Kompresi

Ada beberapa metode kompresi, metode kompresi dapat dibagi ke dalam tiga kategori, yaitu:

a. Metode *symbolwase*, menghitung peluang kemunculan dari tiap simbol dalam *file input*, lalu mengkodekan satu simbol dalam satu waktu, di mana simbol yang lebih sering muncul diberi kode lebih pendek dibandingkan simbol yang lebih jarang muncul. Contoh: algoritma Huffman.
b. Metode *dictionary*, menggantikan karakter/fragmen dalam *file input* dengan indeks lokasi dari karakter/fragmen tersebut dalam sebuah kamus (*dictionary*). Contoh: algoritma LZW.
c. Metode *predictive*, menggunakan model *finite-context* atau *finite-state* untuk memprediksi distribusi probabilitas dari simbol-simbol selanjutnya. Contoh: algoritma DMC.

Jenis kompresi data berdasarkan mode penerimaan data oleh manusia adalah:

a. *Dialogue Mode*: yaitu proses penerimaan data di mana pengirim dan penerima seakan berdialog (*real time*), seperti pada contoh *video conference*. Di mana kompresi data harus berada dalam batas penglihatan dan pendengaran manusia. Waktu tunda (*delay*) tidak boleh lebih dari 150 *Ms*, di mana 50 *Ms* untuk proses kompresi, 100 *Ms* mentransmisikan data dalam jaringan.
b. *Retrieval Mode*: yaitu proses penerimaan data tidak dilakukan secara *realtime*. Dapat dilakukan *fast forward* dan *fast rewind* di *client*. Dapat dilakukan *random access* terhadap data dan dapat bersifat interaktif.

Terdapat beberapa faktor yang sering menjadi pertimbangan dalam memilih suatu metode kompresi yang tepat karena tidak ada suatu metode kompresi yang paling efektif untuk semua jenis *file*. Faktor-faktor tersebut adalah:

a. Kualitas data hasil enkoding: ukuran lebih kecil, data tidak rusak untuk kompresi *lossy*.
b. Kecepatan, *ratio*, dan efisiensi proses kompresi.

II.5. Multimedia

a. Pengertian Multimedia

Istilah Multimedia berasal dari kata multi yang berarti banyak atau bermacam-macam dan kata media yang berarti sarana yang dipakai untuk menyampaikan sesuatu atau alat untuk mendistribusikan dan mempresentasikan informasi.

Multimedia dapat diartikan sebagai penggunaan beberapa media yang berbeda untuk menggabungkan dan menyampaikan informasi dalam bentuk teks, audio, grafik, animasi dan video.

b. Unsur-unsur Multimedia

Unsur-unsur dari Multimedia yaitu :

1. Audio

Audio adalah segala suatu yang dapat didengar. Audio atau suaradalam komputer diolah oleh sound card dari bentuk analog digital. Audio sangat berguna memberi tekanan dalam sebuah adegan atau memberikan efek suara dalam sebuah karya multimedia

2. Gambar / Grafik

Gambar merupakan kumpulan dari banyak tittik yang tersusun sedemikian rupa, sehingga menjadi suatu bentuk yang diinginkan. Gambar merupakan bentuk yang disajikan sebagai sarana yang mudah dipahami dan dimengerti oleh para pemakai. Gambar juga bisa sebagai alat penerjemah.

3. Teks

Tampilan dalam bentuk teks pada program multimedia sangat berperan memberikan kemudahan bagi pemakaiuntuk menyampaikan

suatu informasi. Teks juga sangat berguna untuk menjelaskan adegan yang sedang berlangsung dalam sebuah sistem multimedia. Teks juga memberikan warna tersendiri bagi multimedia.

4. Animasi

Animasi adalah paparan urutan yang setiap tahunya terdapat sedikit perbedaan untuk menghasilkan satu pergerakan secara berterusan. Animasi merupakan satu teknologi yang membolehkan image pengguna kelihatan seolah-olah hidup, dapat bergerak, beraksi dan bercakap.

5. Video

Video adalah sistem gambar hidup atau gambar bergerak yang saling berurutan. Terdapat dua macam video yaitu video analog dan video digital. Video analog dibentuk dari deretan sinyal elektrik (gelombbang analog) yang direkam oleh kamera dan dipancarluaskan melalui gelombang udara. Sedangkan videi digital dibentuk dari sederetan sinyal digital yang berbentuk, yang menggambarkan titik sebagai rangkaian nilai minimum atau maksimum, nilai minimum berarti 0 dan nilai maksimum berarti 1.

c. Pengertian Komputer Multimedia

Multimedia komputer adalah sebuah komputer yang dioptimalkan untuk kinerja multimedia yang tinggi, sehingga memungkinkan bagi seseorang untuk dapat belajar sekaligus menggali banyak informasi. (*Sri Maryati ; IJCS ; Vol 10 No 1 ; 2013; Hal.48*).

II.5.1. Multimedia Pembelajaran

Pembelajaran interaktif adalah pembelajaran yang melibatkan interaksi-interaksi baik antar mahasiswa, mahasiswa dengan dosen, mahasiswa dengan lingkungan atau bahan pembelajaran lainnya. Interaksi adalah elemen substansial dari suatu aktivitas pembelajaran. Interaksi, khususnya bagi mahasiswa, harus diciptakan dan diberi peluang seluas-luasnya sehingga tujuan pembelajaran yang dikehendaki, khususnya mahasiswa dapat tercapai melalui suatu proses interaksi tertentu. (*Rudi Hendrawansyah , Jurnal Teknologi Informasi , Volume 5 Nomor 2, Hal 713*).

II.5.2. Pembelajaran Berbasis Teknologi

Pembelajaran menggunakan teknologi multimedia interaktif klasikal tidak berperan dalam merangsang kemandirian siswa dalam belajar dikelas tapi peranan itu jelas membantu jika program tersebut dapat dipelajari mandiri di luar jam pelajaran., karena mempengaruhi cara berpikir siswa dalam mengolah pelajarannya menjadi lebih dipahami (*Fransisca Tapilouw , Jurnal Pendidikan Teknologi Informasi dan Komunikasi , Vol 1 Nomor 2 , Hal 713*).

II.5.3. Konsep Pembelajaran Berbasis Multimedia

Multimedia pembelajaran adalah multimedia interaktif yang digunakan dalam proses pembelajaran. Lebih berorientasi ke konten termasuk di dalamnya interaktifitas, grafis, sound dan berbagai teknik untuk membantu memahamkan ke anak didik dengan cepat (*Romi Satria Wahono, 2008).Kozma (Kozma, 1991, Dalam : Joko Sutrisno, 2009*) berpendapat bahwa multimediadapat meningkatkan kegiatan belajar.

Media dapat membantu membuat model mental yang lebih baik sehingga membantu pemahaman seorang pembelajar. Menurut pendapat Joko

Sutrisno mengenai multimedia dalam pembelajaran adalah sebagai berikut.

a. Multimedia dapat digunakan untuk membantu pembelajar membentuk model mental yang akan memudahkannya memahami suatu konsep
b. Pemanfaatan multimedia dapat membangkitkan motivasi belajar para pembelajar karena adanya multimedia membuat presentasi pembelajaran lebih menarik. Menurut pendapat Bovee mengenai multimedia pembelajaran adalah sebuah alat yang berfungsi untuk menyampaikan pesan pembelajaran (*Bovee, 1997*). Pembelajaran adalah sebuah proses komunikasi antara pembelajar, pengajar dan bahan ajar. Komunikasi tidak akan berjalan tanpa bantuan sarana penyampai pesan atau media. Bentukbentuk stimulus bisa dipergunakan sebagai media diantaranya adalah hubungan atau interaksi manusia; realita; gambar bergerak atau tidak; tulisan dan suara yang direkam. Kelima bentuk stimulus ini akan membantu pembelajar mempelajari bahasa asing. Namun demikian tidaklah mudah mendapatkan kelima bentuk itu

dalam satu waktu atau tempat.

Multimedia pembelajaran yang baik harus memenuhi beberapa syarat. Multimedia pembelajaran harus meningkatkan motivasi pembelajar. Penggunaan Multimedia pembelajaran mempunyai tujuan memberikan motivasi kepada pembelajar. Selain itu media juga harus merangsang pembelajar mengingat apa yang sudah dipelajari selain memberikan rangsangan belajar baru. Multimedia pembelajaran yang baik juga akan mengaktifkan pembelajar dalam memberikan tanggapan, umpan balik dan juga mendorong siswa untuk melakukan praktek-praktek dengan benar. (*Suraji, Seminar Riset Unggulan Nasional Informatika dan Komputer , Vol 2 No 1 , 2013 , Hal 44*)

II.6. 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

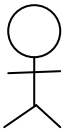
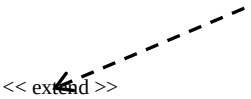
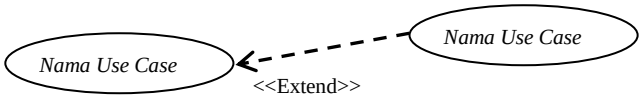
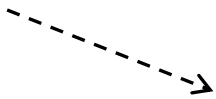
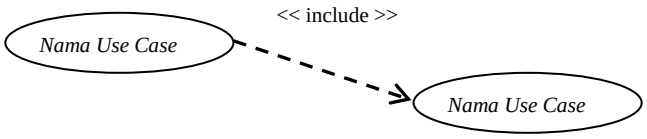
bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), 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.

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 ; S.T, M.Kom ; 2013 : 41)*

Tabel II.2. Simbol *Use Case*

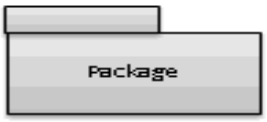
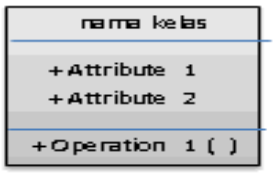
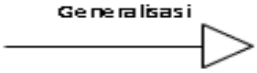
Simbol	**Deskripsi**
Use Case 	Fungsional yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit dan aktor, biasanya menggunakan kata kerja di awal frase nama *use case*
Aktor  *Nama Aktor*	Orang atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat sistem itu sendiri. Jadi walaupun simbol aktor adalah gambar orang, tapi aktor belum tentu merupakan orang. Biasanya menggunakan kata benda di awal frase nama actor
Asosiasi / Association	Komunikasi antar aktor dan *use case* yang berpartisipasi *use case* atau *use case* berinteraksi dengan aktor.
Extend 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walupun tanpa *use case* tambahan itu, mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *use case* tambhan 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; 

Sumber : Yuni Sugiarti, S.T, M.Kom (2013 : 42)

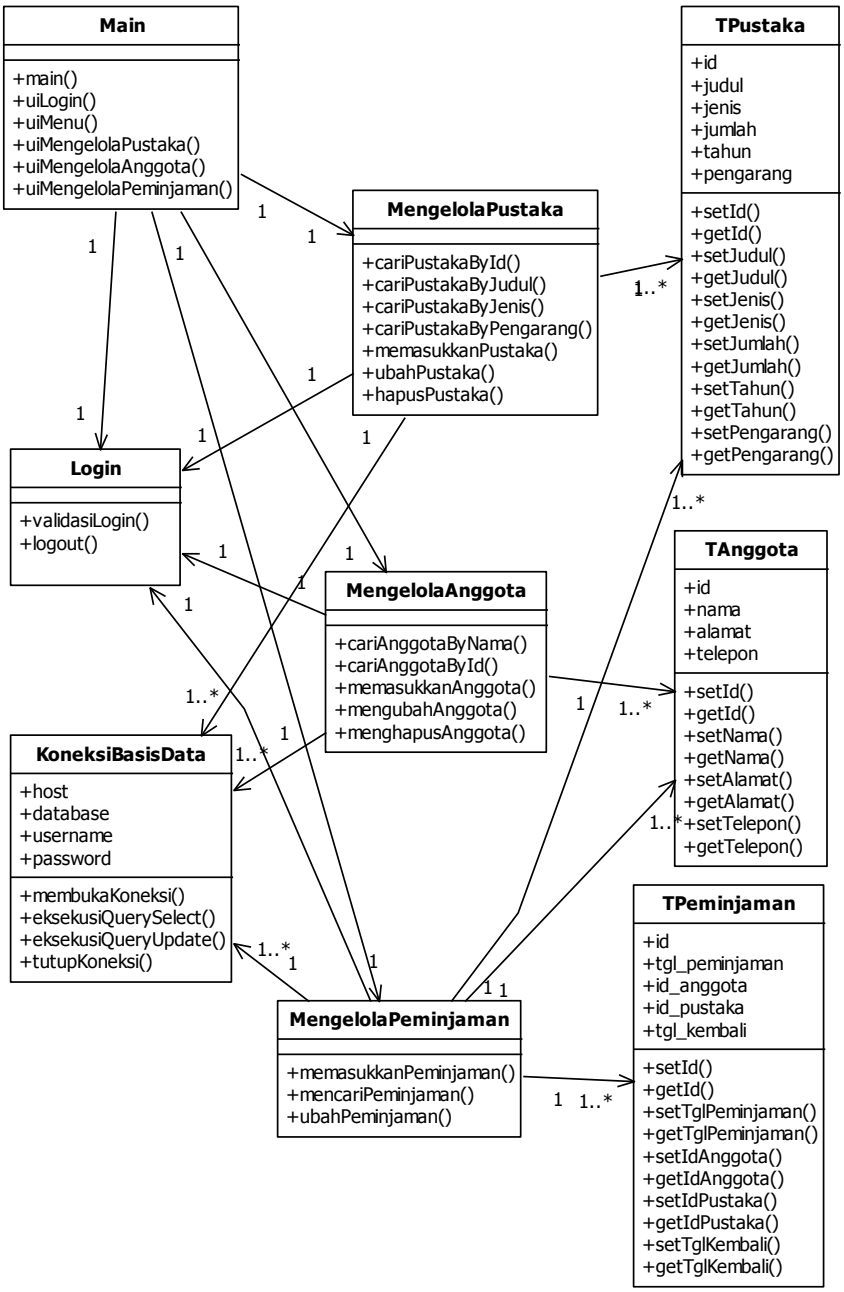
2. *Class Diagram*

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  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.1. *Class* Diagram
Sumber : Yuni Sugiarti, S.T, M.Kom (2013 : 59)



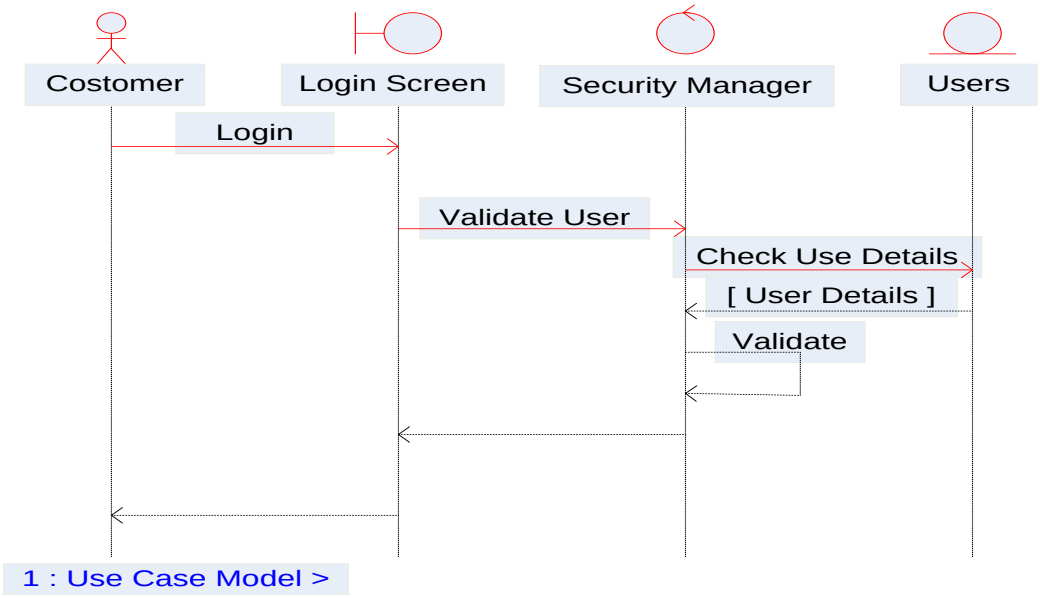
Gambar II.2. Contoh Class Diagram

Sumber : Yuni Sugiarti, S.T, M.Kom (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.3. Contoh *Sequence* Diagram
Sumber : Yuni Sugiarti, S.T, M.Kom (2013 : 63)

4. *Activity Diagram*

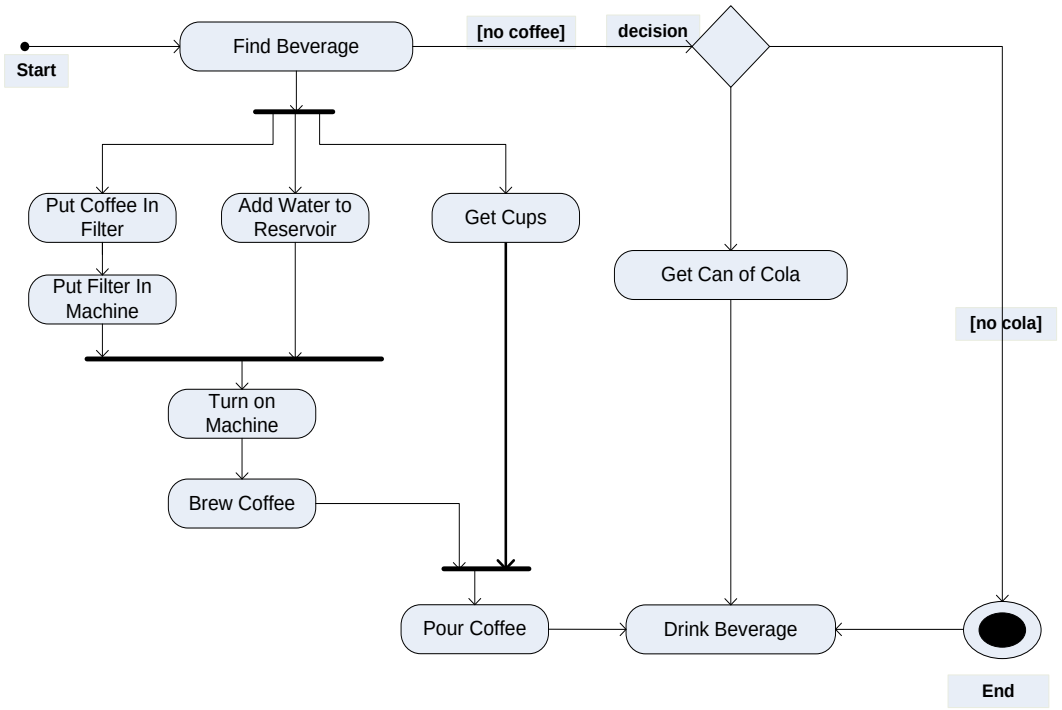
Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

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.4. Activity Diagram
Sumber : Yuni Sugiarti, S.T, M.Kom (2013 : 76)

II.7. *Visual Studio*

Visual Basic.Net (VB.NET) atau biasa disebut *Visual Basic* adalah teknologi permograman *Microsoft* yang dapat digunakan untuk membuat aplikasi dilingkuangn kerja berbasis *Windows*. *Visual Basic.Net* adalah pengembangan dari *Visual Basic* sebelumnya. kelebihan VB.Net terletak pada tampilannya yang lebih canggih dibandingkan edisi *Visual Basic* sebelumnya. selain memiliki kelebihan, VB.Net juga memiliki kekurangan. Kekurangan VB.Net yang terlihat jelas adalah beratnya aplikasi ini apabila dijalankan dikomputer yang memiliki spesifikasi sederhana. *Andi* (2006 : 1)

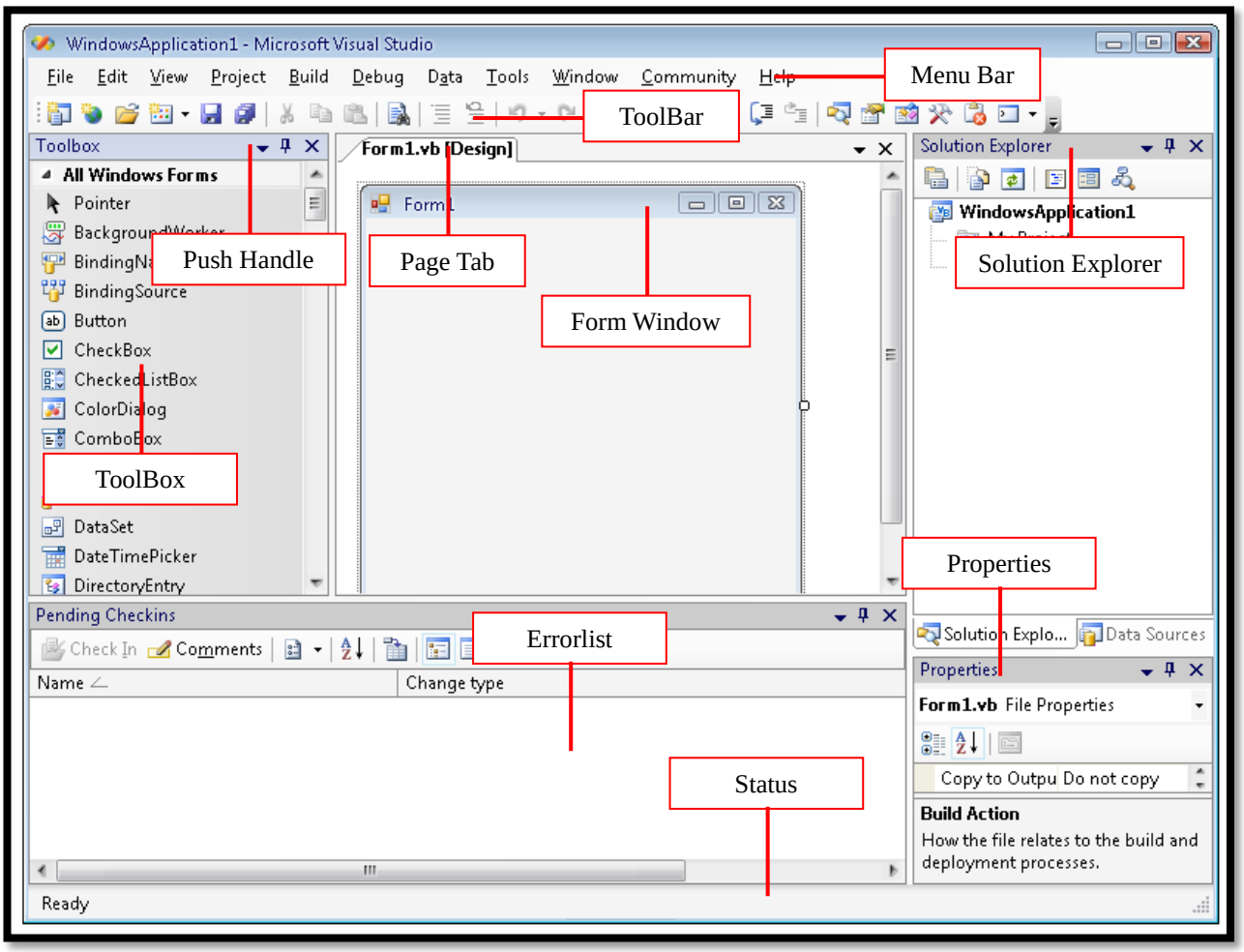
Terdapat beberapa fungsi dasar dalam bahasa pemrograman pada umumnya yang dapat dijelaskan sebagai berikut :

Variable merupakan tempat yang digunakan untuk menampung nilai yang bersifat sementara. Nilai yang diisikan harus sesuai dengan tipe data dari *variable* tersebut, Hal-hal yang perlu diperhatikan dalam mendeklarasikan sebuah *variable* sebagai berikut.

1. Harus terdiri atas huruf, angka atau garis bawah (*underscore*).
2. Harus diawali dengan huruf.
3. Tidak boleh menggunakan kata yang sudah dipakai oleh *Visual Studio* 2005.
4. Panjang *Variable* maksimal 255 karakter.

Konstanta merupakan *variable* yang nilainya bersifat tetap (tidak berubah). Anda hanya perlu sekali dalam mendeklarasiannya. Apabila konstanta menggunakan private berarti konstanta tersebut hanya bias digunakan dalam modul tersebut atau tempat dimana konstanta itu dideklarasikan. Tetapi apabila konstanta menggunakan public, berarti kontanta dapat digunakan diluar modul atau diluar tempat dimana konstanta itu di deklarasikan.

Adapun tampilan *interface Visual Studio.Net* 2005 dapat dilihat pada gambar II.7:



Gambar II.5. Tampilan *Visual Studio.Net*
Sumber : Andi (2006 : 8)

1. *Menu Bar*, Menu standar pada *Visual Basic.Net* atau *Visual Studio* 2005
2. *Toolbar*, Daftar *tool* (perangkat) untuk menjalankan perintah yang sering digunakan
3. *Toolbox*, Daftar kontrol yang dapat ditambahkan ke dalam program sebagai antarmuka (*interface*)
4. *Form Designer*, Digunakan untuk mengedit tampilan *form* serta mengatur posisi kontrol pada *form*

5. *Solution Eplorer*, Digunakan untuk mengolah *file* dan proyek yang berhubungan dengan *Solution Explorer*.
6. *Properties,* Digunakan untuk mengedit properti dari *form* dan kontrol yang sedang di*edit*.
7. *Error List*, Menampilkan pesan *error* jika ada kesalahan.
8. Status, Menampilkan status aplikasi saat dijalankan.