BAB II

TINJAUAN PUSTAKA

II.1. Pengertian Sistem

Sistem (*system*) dapat didefinisikan dengan pendekatan prosedur dan dengan pendekatan komponen. Dengan pendekatan prosedur, sistem dapat didefinisikan sebagai kumpulan dari prosedur-prosedur yang mempunyai tujuan tertentu. Contoh sistem yang didefinisikan dengan pendekatan prosedur ini adalah sistem akuntansi. Sistem ini didefinisikan sebagai kumpulan dari prosedur-prosedur penerimaan kas, pengeluaran kas, penjualan, pembelian dan buku besar.

Dengan pendekatan komponen, sistem dapat didefinisikan sebagai kumpulan dari komponen yang saling berhubungan satu dengan yang lainnya membentuk satu kesatuan untuk mencapai tujuan tertentu. Contoh sistem ini didefinisikan dengan pendekatan ini misalnya adalah sistem komputer yang didefinisikan sebagai kumpulan dari perangkat keras dan perangkat lunak. (Prof. Dr. Jogiyanto HM: 2005; 34)

II.1.1. Karakteristik Sistem

Sistem mempunyai beberapa karakteristik atau sifat-sifat tertentu, antara lain:

1. Komponen Sistem (*Component*)

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang saling bekerja sama membentuk suatu komponen sistem atau bagian-bagian dari sistem.

2. Batasan Sistem (*Boundary*)

   Merupakan daerah yang membatasi suatu sistem dengan sistem yang lain atau dengan lingkungan kerjanya.

3. Subsistem

   Bagian-bagian dari sistem yang beraktifitas dan berinteraksi satu sama lain untuk mencapai tujuan dengan sasarannya masing-masing.

4. Lingkungan Luar Sistem (*Environment*)

   Suatu lingakungan yang ada di luar dari batas sistem yang dipengaruhi oleh operasi sistem.

5. Penghubung Sistem (*Interface*)

   Media penghubung antara suatu subsistem dengan subsistem lain. Adanya penghubung ini memungkinkan berbagai sumber daya mengalir dari suatu subsistem ke subsistem lainnya.

6. Masukan Sistem (*Input*)

   Energi yang masuk kedalam sistem, berupa parawatan dan sinyal. Masukan perawatan adalah energi yang dimasukan supaya sistem tersebut dapat berinteraksi.

7. Keluaran Sistem (*Output*)

   Hasil energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan.

8. Pengolahan Sistem (*Process*)

   Suatu sistem dapat mempunyai suatu bagian pengolah yang akan mengubah masukan menjadi keluaran.

9. Sasaran Sistem (*Object*)

   Tujuan yang ingin dicapai oleh sistem, akan dikatakan berhasil apabila mengenai sasaran atau tujuan. (Kusrini-Andri Koniyo: 2007; 6-7)

II.2. Pengertian Informasi

Informasi adalah data yang sudah diolah menjadi sebuah bentuk yang berarti bagi pengguna, yang bermanfaat dalam pengambilan keputusan saat ini atau mendukung sumber informasi. Data belum memiliki nilai, sedangkan informasi sudah memiliki nilai. Informasi dikatakan bernilai bila manfaatnya lebih besar dibanding biaya untuk mendapatkannya. (Kusrini-Andri Koniyo: 2007; 7-8)

II.2.1. Kualitas Informasi

Informasi yang berkualitas memiliki 3 kriteria, yaitu:

1. Akurat (*accurate*)

   Informasi harus bebas dari kesalahan, tidak bias ataupun menyesatkan. Akurat juga berarti bahwa informasi itu harus dapat dengan jelas mencerminkan maksudnya.

2. Tepat pada waktunya (*timeliness*)

   Informasi yang datang pada penerima tidak boleh terlambat. Di dalam pengambilan keputusan, informasi yang sudah usang tidak lagi bernilai. Bila informasi datang terlambat sihingga pengambilan keputusan terlambat dilakukan, hal itu dapat berakibat fatal bagi perusahaan.

3. Relevan (*relevance*)

   Informasi yang disampaikan harus mempunyai keterkaitan dengan masalah yang akan dibahas dengan informasi tersebut. Informasi harus bermanfaat bagi pemakainya. Di samping karakteristik, nilai informasi juga ikut menentukan kualitasnya. Nilai informasi (*value of information*) ditentukan oleh dua hal, yaitu manfaat dan biaya untuk mendapatkannya. Suatu informasi dikatakan bernilai bila manfaatnya lebih besar dibanding biaya untuk mendapatkannya. (Kusrini-Andri Koniyo: 2007; 8)

II.3. Pengertian Sistem Informasi

Untuk menghasilkan informasi yang berkualitas maka dibuatlah sistem informasi. Sistem informasi didefinisikan oleh Robert A. Laitach dan K. Roscoe Bavis sebagai berikut: “Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan.”

Definisi umum sistem informasi adalah : “Sebuah sistem yang terdiri atas rangkaian subsistem informasi terhadap pengolahan data untuk menghasilkan

informasi yang berguna dalam pengambilan keputusan.” (Kusrini-Andri Koniyo: 2007; 8-9)

II.3.1. Komponen Sistem Informasi

Dalam suatu sistem informasi terdapat komponen-komponen sebagai berikut:

1. Perangkat keras (*hardware*), mencakup berbagai piranti fisik seperti komputer dan printer.
2. Perangkat lunak (*software*) atau program, yaitu sekumpulan instruksi yang memungkinkan perangkat keras memproses data.
3. Prosedur, yaitu sekumpulan aturan yang dipakai untuk mewujudkan pemrosesan data dan pembangkitan keluaran yang dikehendaki.
4. Orang, yaitu semua pihak yang bertanggung jawab dalam pengembangan sistem informasi, pemrosesan dan penggunaan keluaran sistem informasi.
5. Basis data (*database*), yaitu sekumpulan tabel. Hubungan dan lain-lain yang berkaitan dengan penyimpanan data.

Jaringan komputer dan komunikasi data, yaitu sistem penghubung yang memungkinkan sumber (*resource*) dipakai secara bersama atau diakses oleh sejumlah pemakai. (Kusrini-Andri Koniyo: 2007; 8-9)

II.4. Sistem Informasi Akuntansi (SIA)

Sistem informasi akuntansi merupakan sebuah sistem informasi yang mengubah data transaksi bisnis menjadi informasi keuangan yang berguna bagi pemakainya. (Kusrini-Andri Koniyo: 2007; 10-11)

Tujuan dari sistem informasi akuntansi adalah:

1. Mendukung operasi sehari-hari.
2. Mendukung pengambilan keputusan manajemen.
3. Memenuhi kewajiban yang berhubungan dengan pertanggung jawaban.

Komponen-komponen yang terdapat dalam sistem informasi akuntansi adalah sebagai berikut:

1. Orang-orang yang mengoperasikan sistem tersebut.
2. Prosedur-prosedur, baik manual maupun yang terotomatisasi, yang dilibatkan dalam pengumpulan, pemrosesan dan penyimpanan data aktifitas-aktifitas organisasi.
3. Data tentang proses-proses bisnis.
4. *Software* yang dipakai utnuk memproses data organisasi
5. *Insfrastruktur* teknologi informasi.

Sementara itu aktifitas utamanya adalah:

1. *Indbound Logistics* : penerimaan, penyimpanan dan distribusi bahan masukan.
2. Operasi : aktifitas untuk mengubah masukan menjadi barang atau jasa.
3. *Outbound Logistics* : distribusi produk ke pelanggan.
4. Pemasaran dan Penjualan.

5. Pelayanan dukungan purna jual dan *maintenance*.

II.4.1. Penyusutan Aktiva Tetap

Semua penyusutan jenis aktiva tetap, kecuali tanah, akan makin berkurang kemampuannya untuk memberikan jasa bersamaan dengan berlalunya waktu. Beberapa faktor yang mempengaruhi menurunnya kemampuan ini adalah pemakaian, kehausan, ketidakseimbangan kapasitas yang tersedia dengan yang diminta dan keterbelakangan teknologi. Berkurangnya kapasitas berarti berkurangnya nilai aktiva tetap yang bersaangkutan. Hal ini perlu dicatat dan dilaporkan. "Pengakuan adanya penurunan aktiva tetap berwujud disebut penyusutan (*depreciation*)". Penyusutan dapat dihitung tiap-tiap bulan atau ditunda sampai dengan akhir tahun. Apabila dibuat laporan keuangan *interim* secara bulanan, penyusutan yang dilakukan bulanan akan lebih dapat mencerminkan posisi keuangan dan hasil usaha perusahaan dalam bulan yang bersangkutan. (Soemarso S. R: 2005; 24)

II.4.2. Metode Saldo Menurun (*Declining Balance Method)*

Metode garis lurus menganggap bahwa beban penyusutan akan merata sepanjang umur aktiva tetap. Dalam metode saldo menurun, beban penyusutan makin menurun dari tahun ke tahun. Pembebanan yang makin menurun didasarkan pada anggapan bahwa semakin tua kapasitas aktiva tetap dalam memberikan jasanya, juga akan makin menurun. Dalam metode saldo menurun, beban penyusutan dihitung dengan rumus sebagai berikut:

{ (100%/umur ekonomis) x 2 } x Nilai Perolehan/Nilai Buku

Sebagai Contoh:

CV. Matahari Fajar membeli peralatan pada tanggal 3 Januari 2007 seharga Rp. 50.000.000,- dengan nilai sisa diperkirakan sebesar 5% dari harga perolehan. Umur ekonomis 4 tahun (nilai sisa tidak digunakan hanya jebakan saja).

Penyelesaian:

Depresiasi 2007 ={ (100% /4) x 2 } x Rp. 50.000.000 = Rp. 25.000.000,-

Jurnal pada tanggal 31 Desember 2007 :

D : Beban Depresiasi-Peralatan= Rp. 25.000.000,-

K : Akumulasi Depresiasi-Peralatan= Rp. 25.000.000

Depresiasi 2008 = 50% x (Rp. 50 jt – 25 jt) = Rp. 12.500.000

Jurnal pada tanggal 31 Desember 2008 :

D : Beban Depresiasi-Peralatan= Rp. 12.500.000

K : Akumulasi Depresiasi-Peralatan= Rp. 12.500.000

Depresiasi 2009 = 50% x (Rp 50 jt-25jt-12,5jt) = Rp. 6.250.000

Jurnal pada tanggal 31 Desember 2009 :

D : Beban Depresiasi-Peralatan= Rp. 6.250.000

K : Akumulasi Depresiasi-Peralatan= Rp. 6.250.000

Depresiasi 2010 = Rp.50 jt – 25jt-12,5jt-6,25jt = Rp. 6.250.000

Jurnal pada tanggal 31 Desember 2010 :

D : Beban Depresiasi-Peralatan= Rp. 6.250.000

K : Akumulasi Depresiasi-Peralatan= Rp.6.250.000

II.5. Basis Data (*Database*)

Database atau basis data adalah sekumpulan data yang memiliki hubungan secara logika dan diatur dengan susunan tertentu serta disimpan dalam media penyimpanan komputer. Data itu sendiri adalah *representasi* dari semua fakta yang ada pada dunia nyata. *database* sering digunakan untuk melakukan proses terhadap data-data tesebut untuk menghasilkan informasi tertentu. misalnya dari data nama siswa yang berulang tahun pada hari ini. Tentu saja informasi tersebut akan anda dapatkan dari *software* pemroses *database* dengan cara anda memberikan perintah dalam bahasa tertentu yaitu *SQL* (*Structured Query Language*). (Wahana Komputer: 2010; 24)

Pada era kemajuan teknologi seperti sekarang ini, nilai informasi sangatlah penting, terlebih bagi kemajuan perusahaan. Oleh karena itu penggunaan dan penguasaan *database* sangat penting. Dalam *database* ada sebutan-sebutan untuk satuan data yaitu:

1. Karakter, ini adalah satuan data terkecil. data terdiri atau susunan karakter yang pada akhirnya mewakili data yang memiliki arti dari sebuah fakta.
2. *Field*, adalah kumpulan dari karakter yang mewakili fakta tertentu misalnya seperti nama siswa, tanggal lahir, dan lain-lain. Dalam dunia perancangan

database, *field* juga disebut atribut. Bila dipandang dari sudut pemrogaman berorientasi obyek maka *name* dan properti *type*. Properti *name* atau nama adalah properti dari *field* yang berisi *field* yang mewakili data sejenis yang disimpannya. Sedangkan properti *type* adalah properti yang mengatur tipe data dari data yang akan ditampungnya. Misalnya nama *field*nya adalah nama siswa maka tipe datanya adalah *char*, bila nama *field*nya adalah tanggal lahir maka tipe datanya adalah *date*. *Field* dilihat seperti kolom.

3. *Record*, adalah kumpulan dari *field*. Pada *record* anda dapat menemukan banyak sekali informasi penting dengan cara mengkombinasikan *field-field* yang ada.
4. Tabel, adalah sekumpulan dari *record-record* yang memiliki kesamaan *entity* dalam dunia nyata. Kumpulan dari tabel adalah *database*, wujud fisik sebuah *database* dalam komputer adalah sebuah *file* yang didalamnya terdapat berbagai tingkatan data yang telah disebutkan di atas.
5. *File*, adalah bentuk fisik dari penyimpanan data. *File database* berisi semua data yang telah disusun dan diorganisasikan sedemikian rupa sehingga memudahkan pemberian informasi. (Wahana Komputer: 2010; 24 - 25)

II.5.1. Kamus Data

Kamus data (*data dictionary)* digunakan untuk memperjelas aliran data yang digambarkan pada *DFD*. Kamus data adalah kumpulan daftar elemen data yang mengalir pada sistem perangkat lunak sehingga masukan *(input)* dan keluaran *(output)* dapat dipahami secara umum (memiliki standart cara penulisan).

Kamus data biasanya berisi :

1. Nama – nama dari data
2. Digunakan pada – merupakan proses-proses yang terkait data
3. Deskripsi – merupakan deskripsi data
4. informasi tambahan – seperti tipe data, nilai data, batas nilai data, dan komponen yang membentuk data.

Kamus data memiliki beberapa simbol untuk menjelaskan informasi tambahan sebagai berikut :

Tabel II.1 Simbol Kamus Data

Simbol	Keterangan
=	Disusun atau terdiri dari
+	Dan
[|]	Baik…atau…
$\{ \}^n$	n kali diulang / bernilai banyak
( )	Data opsional
…	Batas komentar

(Sumber: Rosa A.S - M.Shalahuddin: 2011; 68)

Kamus data pada *DFD* nanti harus dapat dipetakan dengan hasil perancangan basis data yang dilakukan sebelumnya. Jika ada kamus data yang tidak dapat dipetakan pada tabel hasil perancangan basis data berarti hasil perancangan hasil data dengan perancangan dengan *DFD* masih belum sesuai,

sehingga harus ada yang diperbaiki baik perancangan basis datanya, perancangan *DFD*-nya, atau keduanya. (Rosa A.S-M. Shalahuddin: 2011; 68)

II.5.2. *Entity Relationship Diagram*

Pada dasarnya *ERD* (*Entity Relationship Diagram*) adalah sebuah diagram yang secara konseptual memetakan hubungan antar penyimpanan pada diagram *DFD* di atas. *ERD* ini digunakan untuk melakukan pemodelan terhadap struktur data dan hubungannya. Penggunaannya *ERD* ini dilakukan untuk mengurangi tingkat kerumitan penyusunan sebuah *database* yang baik.

Entity dapat berarti sebuah obyek yang dapat dibedakan dengan obyek lainnya. Obyek tersebut dapat memiliki komponen-komponen data (atribut atau *field*) yang membuatnya dapat dibedakan dari obyek yang lain. Dalam dunia *database entity* memiliki atribut yang menjelaskan karakteristik dari *entity* tersebut. Ada dua macam atribut yang di kenal deskriptif. Hal ini berarti setiap *entity* memiliki himpunan yang diperlukan sebuah *primary key* untuk membedakan anggota-anggota dalam himpunan tersebut.

Atribut dapat memiliki sifat-sifat sebagai berikut:

1. *Atomic*, atomik adalah sifat dari atribut yang menggambarkan bahwa atribut tesebut berisi nilai yang spesifik dan tidak dapat dipecah lagi. Contoh dari sifat atomik adalah *field* status dari tabel karyawan yang hanya berisi menikah atau *single*.
2. *Multivalued*, sifat ini menandakan atribut ini bisa memiliki lebih dari satu nilai untuk tiap *entity* tertentu. Misalnya adalah *field* hobi, hobi dari tiap

karyawan mungkin dan hampir pasti lebih dari satu. Misalnya karyawan A mimiliki hobi membaca , nonton TV dan bersepeda.

3. *Composite*, atribut yang bersifat komposit adalah atribut yang nilainya adalah gabungan dari beberapa atribut yang bersifat atomik. Contohnya adalah atribut alamat yang dapat dipecah menjadi atribut atomik berupa alamat, kode pos, no telepon, dan kota.

Ada beberapa derajat relasi yang dapat terjadi, yaitu :

1. *One to one*, menggambarkan bahwa antara 1 anggota *entity* A hanya dapat berhubungan dengan 1 anggota *entity* B. Biasanya derajat relasi ini digambarkan dengan simbol 1-1.
2. *One to many*, menggambarkan bahwa 1 anggota *entity* A dapat memiliki hubungan dengan lebih dari 1 angota *entity* B. Biasanya derajat relasi ini digambarkan dengan simbol 1-N.
3. *Many to many*, menggambarkan bahwa lebih dari satu anggota A dapat memiliki hubungan dengan lebih dari satu angota *entity* B. Simbol yang digunakan adalah N-N.

II.5.3. *Normalisasi*

Setelah melalui tahapan di atas atau *ERD*, maka hasil pada diagram tersebut mulai direlasasikan pada tabel-tabel *database*. Untuk itu diperlukan sebuah tahapan yang disebut *normalisasi*. *Normalisasi* data adalah proses di mana tabel-tabel pada *database* dites dalam hal saling ketergantungan di antara *field-field* pada sebuah tabel. Misalnya jika pada sebuah tabel terdapat ketergantungan

terhadap lebih dari satu *field* dalam tabel tersebut, maka tabel tersebut harus dipecah menjadi banyak tabel.

Pada proses *normaslisasi* data, aturan yang dijadikan acuan adalah metode ketergantungan fungsional. Teorinya adalah bahwa tiap kolom dalam sebuah tabel selalu memiliki hubungan yang unik dengan sebuah kolom kunci. Misalnya pada sebuah tabel data_siswa ada *field* nomor induk data *field* nama siswa serta *field* tanggal lahir. Maka ketergantungan fungsionalnya dapat dinyatakan sebagai berikut: nmr_induk -> nm_siswa dan nmr_induk -> tgl_lahir. Artinya nm_siswa memiliki ketergantungan fungsional terhadap nmr_induk. *Field* nm_siswa isinya juga ditentukan oleh *field* nmr_induk. Maksud dari semua itu adalah nmr_induk adalah *field* kunci yang menentukan karena tidak ada nomor induk yang sama pada satu sekolah, jadi *field* nmr_induk dapat dijadikan patokan untuk mengisi nm_siswa dan *field* lainnya.

Ada beberapa langkah dalam *normalisasi* tabel, yaitu:

1. *Decomposition* adalah proses mengubah bentuk tabel supaya memenuhi syarat tertentu sebagai tabel yang baik. *Dekomposisi* dapat dikatakan berhasil jika tabel yang dikenal *dekomposisi* bila digabungkan kembali dapat menjadi tabel awal sebelum di *dekomposisi*. *Dekomposisi* akan sering dilakukan dalam proses *normalisasi* untuk memenuhi syarat-syaratnya.
2. Bentuk tidak normal, pada bentuk ini semua data yang ada pada tiap *entity* (diambil atributnya) masih ditampung dalam satu tabel besar. Data yang ada pada tabel ini masih ada yang *redundansi* dan ada juga yang kosong. Semuanya masih tidak tertata rapi.

3. *Normal Form* pertama (*1st Normal Form*), pada tahapan ini tabel di-*dekomposisi* dari tabel bentuk tidak normal yang kemudian dipisahkan menjadi tabel-tabel kecil yang memiliki kriteria tidak memiliki atribut yang bernilai ganda dan *komposit*. Semua atribut harus bersifat atomik.
4. *Normal Form* kedua (*2nd Normal Form*), pada tahapan ini tabel dianggap memenuhi normal kedua jika pada tabel tersebut semua atribut yang bukan kunci primer bergantung penuh terhadap kunci primer tabel tersebut.
5. *Normal Form* ketiga (*3rd Normal Form*), setiap atribut pada tabel selain kunci primer atau kunci utama harus bergantung penuh pada kunci utama. Bentuk normal ketiga biasanya digunakan bila masih ada tabel yang belum efisien. Biasanya penggunaan bentuk normal (*normalisasi*) hanya sampai pada bentuk ketiga, dan tabel yang dihasilkan telah memiliki kualitas untuk membentuk sebuah *database* yang dapat diandalkan. Semua tabel diatas juga telah memenuhi bentuk normal tahap ketiga. (Wahana Komputer: 2010; 30-32)

II.6. *Unified Modelling Language* (UML)

II.6.1. Pengertian UML

Unified Modeling Language (UML) adalah keluarga notasi grafis yang didukung oleh meta-model tunggal, yang membandu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek (OOP).

Bahasa pemodelan grafis telah ada di industri perangkat lunak sejak lama. Pemicu umum dibalik semuanya adalah bahwa bahasa pemrograman berada pada

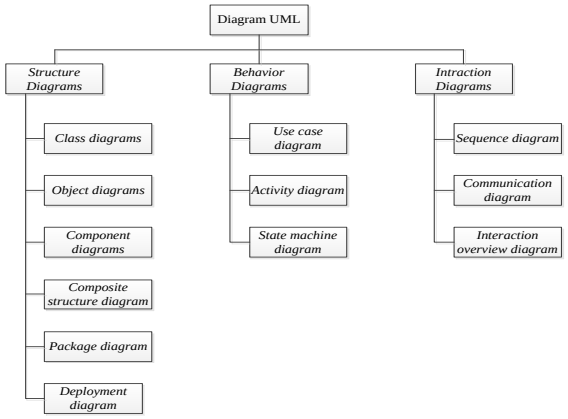
tingkat abstraksi yang tidak terlalu tinggi untuk memfasilitasi diskusi tentang desain.

UML merupakan standar yang relatif terbuka yang dikontrol oleh *Object Manajement Group* (OMG), sebuah *konsorium* terbuka yang terdiri dari banyak perusahaan. OMG dibentuk untuk membuat standar-standar yang mendukung interoperobilitas, khususnya interoperabilitas sistem yang berorientasi objek. OMG mungkin lebih dikenal dengan standar-standar *COBRA* (*Common Object Request Broker Architecture*).

UML lahir dari penggabungan banyak bahasa permodelan grafis berorientasi objek yang berkembang pesat pada akhir 1980-an dan 1990-an. Sejak kehadirannya pada tahun 1997, UML menghancurkan menara Babel tersebut menjadi sejarah. (Martin Fowler ; 2005 : 1 - 2)

II.6.2. Diagram UML

Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar dibawah ini :



Gambar II.1. *Diagram* UML
(Sumber: Rosa A.S-M.Shalahuddin 2011: 121)

Berikut ini penjelasan singkat dari pembagian kategori tersebut :

1. *Structure Diagrams*

Yaitu kumpulan *diagram* yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.

2. *Behavior Diagrams*

Yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.

3. *Interaction Diagrams*

Yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.

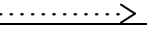
II.6.3. *Class Diagram*

Diagram kelas atau *Class Diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem.

Berikut adalah simbol-simbol yang ada pada diagram kelas.

Tabel II.2. Komponen *Class Diagram*

Simbol/ Arti	Deskripsi
Kelas **Nama_kelas** **+atribut** **+operasi()**	Kelas pada struktur *system*
Antarmuka/*interface* **Nama_interface**	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
Asosiasi/ *association*	Relasi antar kelas dengan makna umum,asosiasi biasanya juga disertai dengan *multiplicity*
Asosiasi berarah/ *directed association*	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/*dependency* 	Relasi antar kelas dengan makna kebergantungan antar kelas

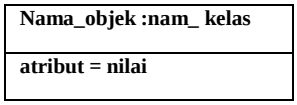
(Sumber: Rosa A.S-M.Shalahuddin: 2011; 122 - 123)

II.6.4. *Object Diagram*

Object Diagram menggambarkan struktur sistem dari penamaan *object* dan jalannya *object* dalam sistem. Pada diagram *object* harus dipastikan semua kelas yang sudah didefinisikan pada diagram kelas harus dipakai objeknya, karena jika tidak pendefinisian kelas itu tidak dapat dipertanggungjawabkan.

Berikut adalah simbol-simbol yang ada pada diagram objek:

Tabel II.3. Komponen *Object Diagram*

Simbol/ Arti	Deskripsi
Kelas 	Objek dari kelas yang berjalan saat dijalankan
Link	Relasi antar objek

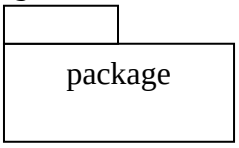
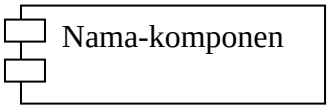
(Sumber: Rosa A.S-M.Shalahuddin: 2011; 124)

II.6.5. *Component Diagram*

Diagram komponen atau *component diagram* dibuat untuk menunjukkan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem.diagram komponen fokus pada komponen sistem yang dibutuhkan dan ada di dalam sistem.

Berikut adalah simbol-simbol yang ada pada diagram komponen:

Tabel II.4. Komponen *Component Diagram*

Simbol/ Arti	Deskripsi
Package 	*Package* merupakan sebuah bungkusan dari satu atau lebih komponen
Komponen 	Komponen system
Kebergantungan/ *dependency* ……	Kebergantungan antar komponen, arah panah mengarah pada komponen yang dipakai
Antarmuka/ *interface*  **Nama_interface**	Sama dengan konsep *interface* dalam pemrograman berorientasi objek agar tidak mengakses langsung komponen
Link	Relasi antar komponen

(Sumber: Rosa A.S-M.Shalahuddin: 2011; 125-126)

II.6.6. *Composite Structure Diagram*

Diagram ini digunakan untuk menggambarkan struktur dari bagian-bagian yang saling terhubung maupun mendeskripsikan struktur pada saat berjalan (*runtime*) dari *instance* yang saling terhubung.

II.6.7. *Package Diagram*

Package diagram menyediakan cara mengumpulkan elemen-elemen yang saling terkait dalam diagram UML.

II.6.8. *Deployment Diagram*

Diagram *deployment* atau *deployment* diagram menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi. (Rosa A.S-M.Shalahuddin: 2011; 127-129)

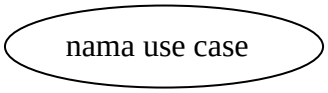
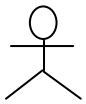
II.6.9. *Use Case Diagram*

Use case atau diagram *use case* merupakan pemodelan untuk kelakukan (behavior) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih *actor* dengan sistem informasi yang akan dibuat.

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami.

Berikut adalah simbol-simbol yang ada pada diagram *use case*:

Tabel II.5. Komponen *Use Case Diagram*

Simbol/ Arti	Deskripsi
Use case 	*Fungsionalitas* yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.
Aktor / *actor*  nama aktor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yanga akan dibuat itu sendiri, jadi malaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang
Asosiasi / *association* 	Komunikasi antar aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan actor
Ekstensi / *extend*	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu

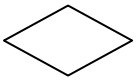
(Sumber: Rosa A.S-M.Shalahuddin: 2011; 130-131)

II.6.10. *Activity Diagram*

Diagram aktifitas atau *activity* diagram menggambarkan *workflow* (aliran kerja) atau aktifitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktifitas menggambarkan aktifitas sistem bukan apa yang dilakukan *actor*, jadi aktifitas yang dapat dilakukan oleh sistem.

Berikut adalah simbol-simbol yang ada pada diagram aktifitas:

Tabel II.6. Komponen *Activity Diagram*

Simbol/ Arti	Deskripsi
Status awal 	Status awal aktifitas sistem,sebuah diagram aktivitas memiliki sebuah status awal
Aktifitas 	Aktifitas yang dilakukan sistem, aktifitas biasanya diawali dengan kata kerja
Percabangan / *decision* 	*Asosiasi* percabangan dimana jika ada pilihan aktifitas lebih dari satu
Penggabungan / *join* 	*Asosiasi* penggabungan dimana lebih dari satu aktifitas digabungkan menjadi satu
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktifitas memiliki sebuah status akhir

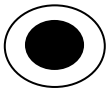
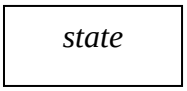
(Sumber: Rosa A.S-M.Shalahuddin: 2011; 134-135)

II.6.11. *State Machine Diagram*

State machine diagram atau diagram mesin statu digunakan untuk menggambarkan perubahan status atau transisi status dari sebuah mesin atau sistem.

Berikut komponen-komponen dasar yang ada dalam *state machine diagram*:

Tabel II.7. Komponen *State Machine Diagram*

Simbol/ Arti	Deskripsi
Start (Initial State) 	Start atau *initial state* adalah *state* atau keadaan awal pada saat sistem mulai hidup
End (*Final State*) 	*End* atau *final state* adalah *state* keadaan akhir dari daur hidup suatu system
Event 	*Event* adalah kegiatan yang menyebabkan berubahnya status mesin
State 	*State* atau status adalah keadaan sistem pada waktu tertentu. *State* dapat berubah jika ada *event* tertentu yang memicu perubahan tersebut

(Sumber: Rosa A.S-M.Shalahuddin: 2011; 136-137)

II.6.12. *Sequence Diagram*

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. (Rosa A.S - M. Shalahuddin: 2011; 137)

II.6.13. *Communication Diagram*

Diagram komunikasi menggambarkan interaksi antar objek/ bagian dalam bentuk urutan pengiriman pesan. Diagram komunikasi mempresentasikan informasi yang diperoleh dari Diagram Kelas, Diagram Sekuen, dan Diagram *Use Case* untuk mendeskripsikan gabungan antar struktur statis dan tingkah laku dinamis dari suatu sistem. (Rosa A.S-M.Shalahuddin: 2011; 140)

II.6.14. *Timing Diagram*

Timing diagram merupakan diagram yang focus pada penggambaran terkait batasan waktu. *Timing diagram* digunakan untuk menggambarkan tingkah laku sistem dalam periode waktu tertentu.

II.6.15. *Interaction Overview Diagram*

Interaction Overview diagram mirip dengan diagram aktivitas yang berfungsi untuk menggambarkan sekumpulan urutan aktivitas. *Interaction overview diagram* adalah bentuk aktivitas diagram yang setiap titik mempresentasikan diagram interaksi. (Rosa A.S-M.Shalahuddin: 2011; 141-143)

II.7. *Database SQL Server*

SQL Server adalah sebuah terobosan baru dari *Microsoft* dalam bidang *database*. *SQL Server* adalah sebuah DBMS (*Database Management System*) yang dibuat oleh *Microsoft* untuk ikut berkecimpung dalam persaingan dunia pengolahan data penyusul pendahulunya seperti IBM dan *Oracle*. *SQL Server* dibuat pada saat kemajuan dalam bidang *hardware* sedemikian pesat. Oleh karena itu sudah dapat dipastikan bahwa *SQL Server* membawa beberapa terobosan dalam bidang pengolahan dan penyimpanan data.

Terdapat beberapa versi *SQL Server* seperti berikut ini:

1. Versi *compact* ini adalah versi “tipis” dari semua versi yang ada. Versi ini seperti versi desktop pada *SQL Server* 2000. Versi ini juga digunakan pada *handheld device* seperti Packet PC, PDA, *Smart Phone*, Tablet PC.

2. Versi *Express* Ini adalah versi “ringan” dari semua versi yang ada (tetapi versi ini berbeda dengan versi *compact*) dan paling cocok untuk latihan para pengembang aplikasi. Versi ini memuat *Express Manager* standar, integrasi dengan CLR dan XML. (Wahana Komputer: 2010; 2-3)

II.8. Pemrograman *Visual Basic* 2008

Visual Basic merupakan salah satu bahasa pemrograman yang andal dan banyak digunakan oleh pengembang untuk membangun berbagai macam aplikasi *Windows*. *Visual Basic* 2008 atau *Visual Basic* 9 adalah versi terbaru yang telah diluncurkan bersama *C#, Visual C++*, dan *Visual Web Developer* dalam satu paket *Visual Studio* 2008.

Visual Basic 2008 merupakan aplikasi pemrograman yang menggunakan teknologi *.NET Framework*. Teknologi *.NET* Framework merupakan komponen *Windows* yang *terintegrasi* serta mendukung pembuatan, penggunaan aplikasi, dan halaman *web*. Teknologi *.NET Framework* mempunyai 2 komponen utama, yaitu CLR (*Common Language Runtime*) dan *Class Library*. CLR digunakan untuk menjalankan aplikasi yang berbasis *.NET*, sedangkan *Library* adalah kelas pustaka atau perintah yang digunakan untuk membangun aplikasi. (Wahana Komputer: 2008; 2)

Visual Basic adalah bahasa pemrograman tingkat tinggi yang sudah sangat terkenal, dimulai dangan *BASIC* yang terdapat pada komputer 'angkatan tua" seperti AT286. Pada saat itu, bahasa *BASIC* merupakan sebuah bahasa yang sangat di andalkan dalam pembuatan beberapa aplikasi penting. *BASIC* digemari

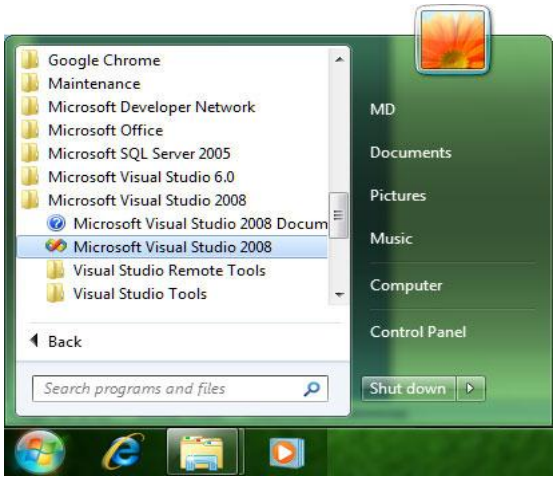
karena susunan programnya yang membebaskan kita untuk melompat dari satu baris program ke baris program yang lainnya. Versi *BASIC* lainnya adalah *BASICA*, *Qbasic*, *Turbo Basic*, dan lain - lainnya. Bahasa *BASIC* terdapat di masa penggunaan sistem operasi *DOS*.

Seiring dengan perkembangan sistem operasi ke sistem berbasis grafik, para penguna *DOS* beralih ke *WINDOWS*. Keberadaan *WINDOWS* mengilhami para programer untuk menciptakan program dengan tampilan grafik yang mirip dengan *WINDOWS*, demikian juga denagn *BASIC* yang juga dikembangkan dengan tambahan beberapa fungsi untuk menciptakan tampilan grafik seperti *WINDOWS*. Kemudahan pemrograman dengan *drag and drop* ini kemudian lebih dikenal dengan *Visual Programing*, metode yang memungkikan programer tidak perlu memasukkan koordinat sebuah jendela lagi secara manual, termasuk pada pengembangannya, programer hanya sedikit sekali memasukkan *coding* (memasukkan kode program), demi terciptanya sebuah program aplikasi yang tangguh.

Visual Basic 2008 disebut juga *Visual Basic* 9. *Visual Basic* ini terdapat dalam produk *Microsoft* yaitu *Visual Stodio* 2008. *Visual Studio* 2008 dan semua anggotanya yang terdiri dari: *Visual Basic*. *Visual C++*, *Visual C#, Visual web developer*, memerlukan *platform* pemrograman yang tepat, yaitu: profesional, *Visual Studio Tools Team Edition*, dan *Express* adalah versi "gratis" dari *Microsoft*. (Wahana Komputer: 2008; 2)

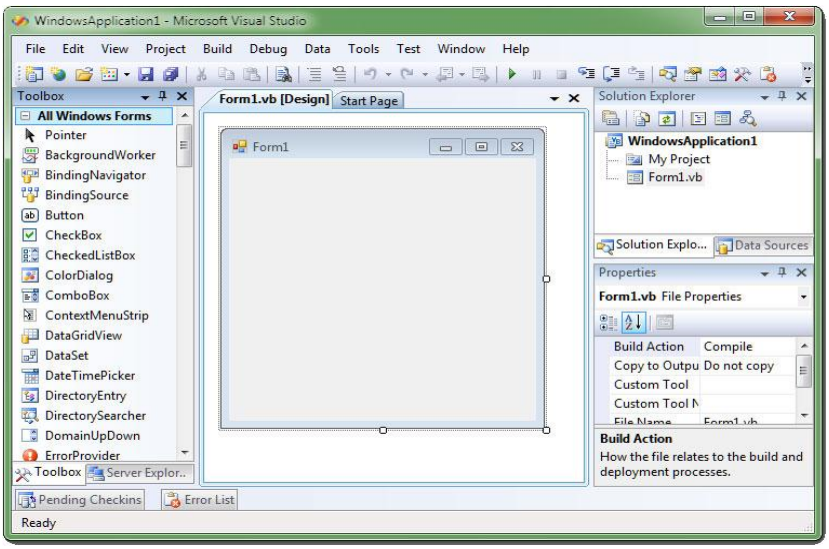
Beberapa langkah yang dapat dilakukan untuk mengoperasikan *Visual Basic* 2008 yaitu :

1. Klik menu *Start* > *All Program* > *Microsoft Visual Studio* 2008 > *Microsoft Visual Studio* 2008.



Gambar II.2. Tampilan Start Menu *Visual Studio* 2008
(Sumber: Wahana Komputer: 2009; 9)

2. Kemudian Klik menu *File* > *New Project*. Lalu pilih *project type* : *Visual Basic* lalu klik OK. Berikut ini tampilan Visual Basic 2008.



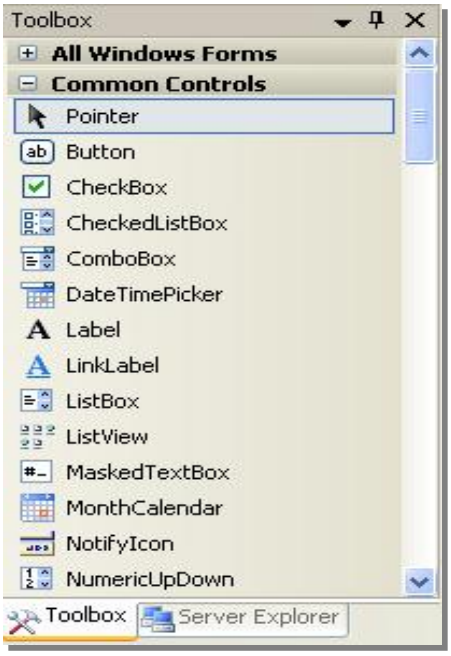
Gambar II.3. Tampilan *Interface Visual Basic* 2008
(Sumber: Wahana Komputer: 2009; 11)

II.8.1. Komponen IDE *Visual Basic* 2008

IDE *Visual Basic* 2008 memiliki beberapa komponen yang akan membantu proses pengembangan program yaitu :

1. *Toolbox*

   Toolbox adalah salah satu yang bisanya digunakan dalam *window*, menyediakan daftar teks perintah, elemen *interface* pengguna, dan *item-item* lainnya yang dapat digunakan dalam *project* yang kita kerjakan.



Gambar II.4. Tampilan Toolbox
(Sumber : Wahana Komputer, 2009:13)

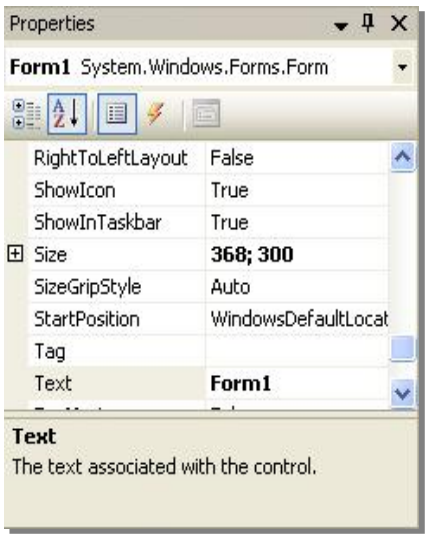
2. *Solution Explorer*

   Pada banyak cara, *solution explorer* mirip dengan fitur *windows explorer* dalam *windows*. Jendela *solution explorer* menampilkan semua objek yang saat itu terbuka, pada grup atau window jendela *solution* berisi *project-project* yang merupakan aplikasi individual.

Gambar II.5. Tampilan Solution Explorer
(Sumber: Wahana Komputer: 2009; 15)

3. *Properties*

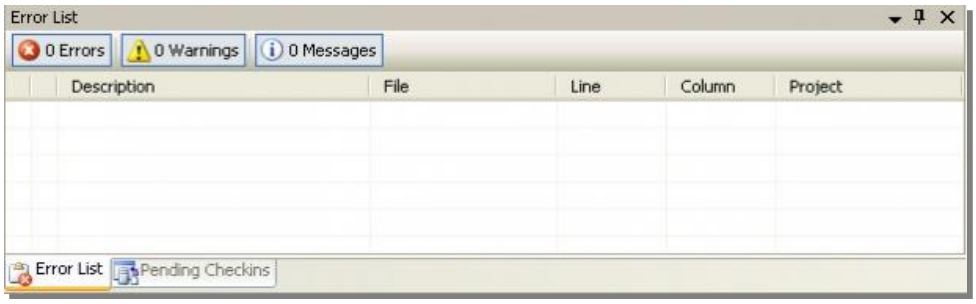
Visual Studio memungkinkan anda untuk bekerja dengan berbagai *item-item*, *project*, solusi, *form*, *class-class*, dan lain-lain. Semuanya memiliki atribut atau *properties*. *Properties* adalah bagian informasi yang menjelaskan *item*, seperti nama *project*.



Gambar II.6. Tampilan *Properties*
(Sumber: Wahana Komputer: 2009; 16)

4. *Error List*

Window ini berfungsi untuk membantu kita dalam mengontrol kesalahan yang terjadi dalam menjalankan program atau menulis program. Setiap kesalahan yang terjadi akan ditampilkan pada *window error list*.



Gambar II.7. Tampilan *Error List*
(Sumber: Wahana Komputer: 2009; 16)

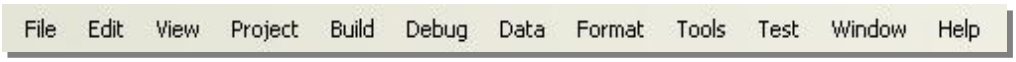
5. Jendela *Design* (*Form*)

Form diibaratkan bagai kanvas yang dapat kita gunakan untuk membuat desain aplikasi yang telah kita rancangkan.

Gambar II.8. Tampilan Jendela Design
(Sumber: Wahana Komputer: 2009; 18)

6. Menu Bar

Seperti pada program-program yang lain, menu disini dapat digunakan sesuai dengan fungsinya masing-masing



Gambar II.9. Tampilan Menu Bar
(Sumber: Wahana Komputer: 2009; 20)

7. *Windows Code*

Windows code digunakan sebagai tempat untuk memberikan perintah atau instruksi suatu program agar dapat berfungsi sesuai dengan yang kita inginkan.

Gambar II.10. Tampilan *Window Code*
(Sumber: Wahana Komputer: 2009; 18)