BAB II

TINJAUAN PUSTAKA

II.1. Sistem

Sistem merupakan sekumpulan elemen-elemen yang saling berintegrasi serta melaksanakan fungsinya masing-masing untuk mencapai tujuan yang telah ditetapkan(Sulindawati dan Muhammad Fathoni,2010).

Menurut Jerry FithGerald, menyatakan bahwa suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan dan berkumpulan bersama-sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sasaran tertentu.

II.1.1. Karakteristik Sistem

Model umum sebuah sistem adalah input,proses, dan output. Hal ini merupakan konsep sebuah sistem yang sangat sederhana sebab sebuah sistem dapat mempunyai beberapa masukan dan keluaran. Selain itu sebuah sistem memiliki karakteristik atau sifat-sifat tertentu,yang mencirikan bahwa hal tersebut bisa dikatakan sebagai suatu sistem. Adapun karakteristik yang dimaksud adalah sebagai berikut:

1. Komponen Sistem

   Suatu sistem terdiri dari sejumlah yang saling berinteraksi,saling bekerja sama membentuk satu kesatuan. Komponen-komponen sistem tersebut dapat berupa suatu bentuk subsistem atau bagian-bagian dari sistem.

2. Batasan Sistem

   Batasan sistem merupakan daerah yang membatasi antara sistem dengan sistem yang lainnya atau sistem dengan lingkungan luarnya.Batasan sistem ini memungkinkan suatu sistem dipandang sebagai suatu kesatuan,karena dengan batasan sistem ini fungsi dari subsistem yang satu dengan lainnya berbeda tetapi tetap saling berinteraksi.Batasan suatu sistem menunjukkan ruang lingkup dari sistem tersebut.

3. Lingkungan Luar Sistem

   Lingkungan luar dari suatu sistem adalah segala sesuatu di luar batas sistem yang memengaruhi operasi dari suatu sistem.Lingkungan luar sistem ini dapat bersifat menguntungkan harus dipelihara dan dijaga agar tidak hilang pengaruhnya,sedangkan lingkungan luar yang merugikan harus dimusnahkan agar tidak mengganggu operasi sistem.

4. Penghubung Sistem

   Penghubung sistem merupakan media penghubung antara subsistem dan subsistem lainnya.Melalui penghubung ini memungkinkan sumber-sumber daya mengalir dari sutu subsistem ke subsistem lainnya.

5. Masukan Sistem

   Masukan Sistem adalah energi yang dimasukan ke dalam sistem. Masukan dapat berupa masukan perawatan(Maintenance Input) dan masukan sinyal(Signal Input). Maintenance input adalah energi yang dimasukkan supaya sistem tersebut dapat beroperasi. Signal input adalah energi yang diproses untuk mendapatkan keluaran.

6. Keluaran Sistem

   Keluaran Sistem adalah hasil energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan.

7. Pengolahan Sistem

   Suatu sistem dapat mempunyai suatu bagian pengolah atau sistem itu sendiri sebagai pengolahnya. Pengolah akan mengubah masukan menjadi keluaran.

8. Sasaran Sistem

   Suatu sistem mempunyai tujuan atau sasaran. Kalau suatu sistem tidak mempunyai sasaran,maka operasi sistem tidak ada gunanya. Sasaran dari sistem sangat menentukan sekali masukan yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem (Sulindawati dan Muhammad Fathoni,2010).

II.1.2. Klasifikasi Sistem

Sistem merupakan suatu bentuk integrasi antara satu komponen dengan komponen lain karena sistem memiliki sasaran yang berbeda setiap kasus yang terjadi yang ada di dalam sistem tersebut. Oleh karena itu sistem dapat diklasifikasikan dari beberapa sudut pandangnya antara lain :

1. Sistem Abstrak dan Sistem Fisik

   Sistem Abstrak adalah sistem yang berupa pemikiran atau ide-ide yang tidak nampak secara fisik.Sedangkan Sistem Fisik merupakan sistem yang ada secara fisik.

2. Sistem Alamiah dan Sistem Buatan Manusia

   Sistem Alamiah adalah sistem yang terjadi melalui sistem alam, tidak dibuat oleh manusia.Sedangkan Sistem Buatan Manusia adalah sistem yang dirancang oleh manusia yang melibatkan antara manusai dengan mesin yang sering disebut dengan *Human machine*.

3. Sistem Tertentu dan Sistem Tak Tentu

   Sistem Tertentu adalah beroperasi dengan tingkah laku yang sudah dapat diprediksi.Sedangkan Sistem Tak Tentu merupakan sistem yang kondisi masa depannya tidak dapat diprediksi karena mengandung unsur probabilitas.

4. Sistem Tertutup dan Sistem Terbuka

Sistem Tertutup merupakan sistem yang tidak berhubungan dan tidak terpengaruh dengan lingkungan luarnya. Sedangkan Sistem Terbuka adalah Sistem yang berhubungan dan terpengaruh dengan lingkungan luarnya (Sulindawati dan Muhammad Fathoni,2010).

II.2. Informasi

Informasi adalah data yang diolah menjadi suatu bentuk yang berarti bagi penerimanya dan bermanfaat dalam pengambilan keputusan pada saat sekarang atau yang akan datang. Informasi juga merupakan fakta-fakta atau data yang telah diproses sedemikian rupa atau mengalami proses transformasi data sehingga berubah bentuk menjadi informasi (Sulindawati dan Muhammad Fathoni,2010).

Adapun Kualitas dari informasi adalah sebagai berikut :

1. Akurat, Berarti informasi harus bebas dari kesalahan-kesalahan dan biasa atau menyesatkan.
2. Tepat Pada Waktunya, Berarti informasi yang pada penerima tidak boleh terlambat.
3. Relevan, Berarti informasi tersebut mempunyai manfaat untuk pemakainya.

II.2.1. Sistem Informasi

Sistem Informasi dapat diartikan sebagai suatu sistem di dalam organisasi yang merupakan kombinasi dari orang-orang,fasilitas, teknologi, media, prosedur-prosedur, dan pengendalian yang ditujukan untuk mendapatkan jalur kombinasi yang penting.

II.2.2. Komponen-Komponen Sistem Informasi

Sistem informasi dari komponen-komponen yang disebut blok bangunan, yang terdiri dari blok masukan, blok model, blok keluaran, blok teknologi, blok basis data dan kendali. Sebagai suatu sistem, keenam blok tersebut masing-masing saling berinteraksi satu dengan yang lain membentuk satu kesatuan untuk mencapai sasaran.

1. Blok Masukan (*Input Blok*)

   Input mewakili data yang masuk ke dalam sistem informasi. *Input*di sini termasuk metode dan media untuk menangkap data yang akan dimasukan, yang dapat berupa dokumen-dokumen dasar.

2. Blok Model (*Model Blok*)

   Blok ini terdiri dari kombinasi prosedur, logika, dan model matematik yang akan memanipulasi data *Input* dan data yang tersimpan di basis data dengan cara yang sudah tertentu untuk menghasilkan keluaran yang diinginkan.

3. Blok Keluaran *(Output Blok*)

   Produk dari sistem informasi adalah keluaran yang merupakan informasi yang berkualitas dan dokumentasi yang berguna untuk semua tingkatan manajemen serta pemakaian sistem.

4. Blok Teknologi (*Technology Blok*)

   Teknologi merupakan " *tool box*" dalam sistem informasi. Teknologi digunakan untuk menerima Input, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan keluaran dan membantu pengendalian dari sistem keseluruhan.Teknologi sistem terdiri dari 3(tiga) bagian yaitu teknisi teknologi (*brainware*), perangkat lunak (*software*), dan perangkat keras (*hardware*).

5. Blok Data

Basis data (database) merupakan kumpulan data yang saling berkaitan dan berhubungan satu dengan lainnya, tersimpang di perangkat keras komputer dan menggunakan perangkat lunak untuk memanipulasinya. Data perlu disimpan di basis data untuk keperluan penyediaan informasi lebih lanjut.Data di dalam basis data perlu diorganisasikan sedemikian rupa supaya informasi yang dihasilkan berkualitas.Organisasi basis data yang

baik juga berguna untuk efesiensi kapasitas penyimpananya. Basis data diakses atau dimanipulasi menggunakan perangkat lunak paket yang disebut DBMS (*database management system).*

6. Blok Kendali (*Control Block*)

   Banyak hal yang dapat merusak sistem informasi, seperti bencana alam, api, temperature, air, debu, kecurangan-kecurangan, kegagalan-kegagalan sistem itu sendiri, ketidak efesienan, sabotase,dan lain sebagainya.

II.3. Entity Relationship Diagram (ERD)

Entity Relationship Model / ER_M merupakan suatu model data yang dikembangkan berdasarkan obyek. ER_M digunakan untuk menjelaskan hubungan antar data dalam basis data kepada pengguna secara logic. ER_M digambarkan dalam bentuk diagram yang disebut diagram ER (*ER_Diagram/*ER_D). untuk menggambarkan ER_D digunakan simbol-simbol grafis tertentu. ER_D berguna untuk memodelkan sistem yang nantinya basis datanya akan dikembangkan.Sebuah diagram ER/ER_D tersusun atas tiga komponen, yaitu entitas, atribut, dan kerelasian antar entitas.

1. Entitas (*Entity*)

   Entitas menunjukan obyek-obyek dasar yang terkait di dalam sistem.Obyek dasar dapat berupa orang, benda, atau hal yang keterangannya perlu disimpan dalam basis data. Untuk menggambarkan sebuah entitas digunakan aturan sebagai berikut:

a. Entitas dinyatakan dengan simbol persegi panjang.
b. Nama entitas dituliskan di dalam simbol persegi panjang.
c. Nama entitas berupa kata benda, tunggal.
d. Nama entitas sedapat mungkin menggunakan nama yang mudah dipahami dan dapat menyatakan maknanya dengan jelas.

Sering kali nama entitas dapat tersusun atas lebih dari satu kata. Untuk memenuhi aturan penggambaran tersebut maka sering digunakan tanda _ (garis bawah/*hypen/underscore*) yang dimaksudkan untuk menyatakan bahwa beberapa kata terrsebut dianggap sebagai kata tunggal.

2. Atribut (*Attribute*)

Atribut sering pula disebut sebagai properti (*property*).Merupakan keterangan-keterangan yang terkait pada sebuah entitas yang perlu disimpan dalam basis data.Atribut berfungsi sebagai penjelas pada sebuah entitas (Sutanta, 2004).

3. Kerelasian Antar Entitas (*Relationship*)

Keserelasian antar entitas mendefenisikan hubungan antara dua buah entitas.Keserelasian adalah kejadian atau transaksi yang terjadi di antara dua buah entitas yang keterangannya perlu disimpan dalam basis data (Martin, 1975). Kejadian atau transaksi yang tidak perlu disimpan dalam basis data (sekalipun benar-benar terjadi) bukan termasuk kerelasian (Edhy Sutanta ; 2011 : 91).

II.4. Normalisasi

Normalisasi adalah teknik perancangan yang banyak digunakan sebagai pemandudalam merancang basis data relasional. Pada dasarnya normalisasi adalah proses dua langkah yang meletakkan data dalam bentuk tabulasi dengan menghilangkan kelompok berulang lalu menghilangkan data yang terduplikasi dari tabel relasional.

1. Bentuk Normal Pertama (1 NF)

   Contoh yang kita gunakan disini adalah sebuah perancangan yang mendapatkan barang dari sejumlah pemasok. Masing-masing pemasok berada pada satu kota. Sebuah kota dapat mempunyai lebih dari satu pemasok dan masing-masing kota mempunyai kode status tersendiri. Masing-masing pemasok bisa menyediakan banyak barang. Tabel Relasional dapat dituliskan sebagai berikut :

   PEMASOK (P#, Status, Kota, b# qty) dimana

   P# : kode pemasok (kunci utama)

   Status : kode status kota

   Kota : nama kota

   B# : barang yang dipasok

   Oty : jumlah barang yang dipasok

   Sebuah tabel relasional secara defenisi selalu berada dalam bentuk normal pertama.Semua nilai pada kolom-kolomnya adalah atomi.Ini berarti kolom-kolom ini tidak mempunyai nilai berulang. Tabel II.2. Menunjukkan tabel pemasok dalam 1 NF (Janner Simarmata, dkk, 2010)

Tabel II.1. Normalisasi Pertama Pemasok

P#	Status	Kota	B#	Oty
P1	20	Yogyakarta	B1	300
P1	20	Yogyakarta	B2	200
P1	20	Yogyakarta	B3	400
P1	20	Yogyakarta	B4	200
P1	20	Yogyakarta	B5	100
P1	20	Yogyakarta	B6	100
P2	10	Medan	B1	300
P2	10	Medan	B2	400
P3	10	Medan	B2	200
P4	20	Yogyakarta	B2	200
P4	20	Yogyakarta	B4	300
P4	20	Yogyakarta	B5	400

(Sumber : Janner Simarmata, dkk, 2010)

2. Bentuk Normal Kedua (2 NF)

Defenisi bentuk normal kedua menyatakan bahwa tabel dengan kunci utama gabungan hanya dapat berada pada 1 NF, tetapi tidak pada 2 NF, Sebuah tabel relasional berada pada bentuk normal kedua jika dia berada pada 1 NF dan setiap kolom bukan kunci yang sepenuhnya tergantung pada kunci utama. Ini berarti bahwa setiap kolom bukan kunci harus tergantung pada seluruh kolom yang membentuk kunci utama.Tabel pemasok berada pada 1 NF, tetapi tidak pada 2 NF karena status dan kota tergantung secara fungsional hanya pada kolom p# dari kunci gabungan (p#, b#). Ini dapat digambarkan dengan membuat daftar ketergantungan fungsional.(Janner Simarmata, dkk, 2010).

P# ⟶ Kota, Status

Kota ⟶ Status

(P#, B#) ⟶ qty

Proses mengubah tabel 1 NF ke 2 NF adalah :

a. Tentukan sembarangan kolom penentuan selain kunci gabungan dan kolom- kolom yang ditentukannya.
b. Buat dan beri nama tabel baru untuk masing-masing penentu dan kolom-kolomyang ditentukan.
c. Pindahkan kolom-kolom yang ditentukan dari tabel asal ke tabel baru penentu akan menjadi kunci utama pada tabel baru.
d. Hapus kolom yang baru dipindahkan dari tabel asal, kecuali penentuan yang akan berfungsi sebagai kunci tamu.
e. Tabel asal bisa diberi nama baru.

   Pada contoh, kita memindahkan kolom p#, status, dan kota ke tabel baru yang disebut pemasok 2. Kolom p# menjadi kunci utama tabel ini. Tabel II.3 menunjukkan hasilnya (Janner Simarmata, dkk, 2010).

Tabel II.2. Tabel Bentuk Normal Kedua (2 NF)

Pemasok 2

P#	Status	Kota
P1	20	Yogyakarta
P2	10	Medan
P3	10	Medan
P4	20	Yogyakarta
P5	30	Bandung

Barang

P#	B#	Oty
P1	B1	300
P1	B2	200
P1	B3	400
P1	B4	200
P1	B5	100
P1	B6	100
P2	B1	300
P2	B2	400
P3	B2	200
P4	B2	300
P4	B4	300

(Sumber : Janner Simarmata, dkk,2010)

3. Bentuk Normal Ketiga (3 NF)

Bentuk normal ketiga mengharuskan semua kolom pada tabel relasional hanya pada kunci utama. Secara definisi, sebuah tabel berada pada bentuk normal ketiga (3NF) jika tabel sudah berada pada 2NF dan setiap kolom yang bukan kunci tidak tergantung secara transistif pada kunci utamanya. Dengan kata lain, semua atribut bukan kunci tergantung secara fungsional hanya pada kunci utama. Tabel barang sudah dalam bentuk normal ketiga. Kolom bukan kunci, qty, tergantung sepenuhnya pada kunci utama(p#, b#). Pemasok masih berada pada 2 NF, tetapi belum berada pada 3 NF karena dia mengandung ketergantungan transitif. Ketergantungan transitif terjadi ketika sebuah kolom bukan kunci, yang ditentukan oleh kunci utama, menentukan kolom lainnya (Janner Simarmata, dkk, 2010).

Konsep ketergantung transitif dapat digambarkan dengan menunjukkan ketergantungan fungsional pada pemasok2, yaitu :

Pemasok2.p# ⟶ Pemasok2, status

Pemasok2.p# ⟶ Pemasok2, kota

Pemasok2,p# ⟶ Pemasok2, status

Perlu dicatat bahtewa pemasok2, status ditentukan, baik oleh kunci utama p#, maupun kolom bukan kunci, kota.

Proses mengunduh tabel menjadi 3 NF adalah :

a. Tentukan semua penentu selain kunci utama dan kolom yang ditentukannya.

b. Buat dan beri nama tabel baru untuk masing-masing penentu dan kolom yang ditentukannya.

c. Pindahkan kolom yang ditentukan dari tabel asal ke tabel baru. Penentu menjadi kunci utama tabel baru.

d. Hapus kolom yang baru saja dipindahkan dari tabel asal, kecuali penentuan yang akan berfungsi sebagai kunci tama.

e. Tabel asal bisa diberi nama baru.

Untuk mengubah PEMASOK2 menjadi 3 NF, kita membuat tabel baru yang disebut KOTA_STATUS dan memindahkan kolom kota dan status ke tabel baru. Status dihapus dari tabel diberi nama baru. Status dihapus dari tabel diberi nama baru PEMASOK_KOTA. Tabel II.4 menunjukkan hasilnya(Janner Simarmata, dkk, 2010).

Tabel II.3. Tabel Bentuk Normal Ketiga (3 NF)

PEMASOK_KOTA

	ota
	ogyakarta
	edan
	edan
	ogyakarta
	ndung

KOTA_STATUS

ota	atus
ogyakarta	
edan	
ndung	
marang	

Sumber : (Janner Simarmata, dkk,2010)

II.4.1.Basis Data(Database)

Basis Data menurut (Raymond McLeod dan George Schell, 2001) Mendefenisikan basis data sebagai keseluruhan data yang disimpan dalam sistem komputer yang menjadi sumber daya organisasi.Kemudian menurut (Abraham Silberschatz, Henry F.Korth, & S.Sudarshan, 2001) merupakan sekumpulan data yang saling berhubungan dan menjadi bagian dari DBMS.

1. Keuntungan DBMS (Database Managemen System)

   DBMS memungkinkan perusahaan maupun pengguna individu untuk :

   a. Mengurangi Kerangkapan Data

      Apabila dibandingkan dengan *file-fie* komputer yang disimpan terpisah di setiap lokasi komputer. DBMS mengurangi jumlah total *file* dengan menghapus data yang terduplikasi di berbagai *file*. Data terduplikasi selebihnya dapat ditempatkan dalam satu *file.*

   b. Mencapai Independensi Data

      Spesifikasi data disimpan dalam skema pada tiap program aplikasi.Perubahan dapat dibuat pada struktur data tanpa mempengaruhi program yang mengakses data.

   c. Mengintegrasikan data beberapa *file*

      Saat file dibentuk sehingga menyediakan kegiatan logis, maka organisasi fisik bukan merupakan kendala. Organisasi logis, pandangan pengguna, dan program aplikasi tidak harus tercermin pada media.

d. Mengambil data dan informasi dengan cepat

   Hubungan-hubungan logis, bahasa manipulasi data, serta bahasa query memungkinkan pengguna mengambil data dalam hitungan detik atau menit.

e. Meningkatkan Keamanan

   DBMS mainframe maupun komputer mikro dapat menyertakan beberapa lapis keamanan seperti kata sandi (*password*), direktori pemakai, dan bahasa sandi (*encryption*) sehingga data yang dikelola akan lebih aman.

2. Kerugian DBMS

   Keputusan menggunakan DBMS mengikut perusahaan atau pengguna untuk :

   a. Memperoleh Perangkat Lunak

      DBMS mainframe masih sangat mahal.Walaupun harga DBMS berbasis komputer mikro lebih murah, tetapi tetap merupakan pengeluaran besar bagi suatu organisasi kecil.

   b. Memperoleh konfigurasi perangkat keras yang besar DBMS sering memerlukan kapasitas penyimpanan kapasitas penyimpanan dan memori lebih besar dari pada program aplikasi lain.

   c. Mempekerjakan dan Mempertahankan staf DBA

      DBMS memerlukan pengetahuan khusu agar dapat memanfaatkan kemampuannya secara penuh.Pengetahuan khusu ini disediakan paling baik oleh para pengguna basis data (DBA). Baik Basis data

terkomputerisasi maupun DBMS bukanlah persyarat untuk memecahkan masalah. Namun, keduanya memberikan dasar-dasar menggunakan komputer sebagai suatu sistem informasi bagi para spesialis informasi dan pengguna.(Janner Simarmata, dkk,2010).

II.4.2. *Unified Modeling Language* (UML)

UML singkatan dari *Unified Modeling Language* yang berarti bahasa pemodelan standart (Chonoles, 2003) mengatakan sebagai bahasa, berarti UML memiliki sintaks dan semantic. Ketika kita membuat model menggunakan konsep UML ada aturan-aturan yang harus diikuti.Bagaimana elemen pada model-model yang kita buat harus berhubungan satu dengan lainnya harus mengikuti standart yang ada.UML bukan hanya sekedar diagram, tetapi juga menceritakan konteksnya. Ketika pelanggan memesan sesuatu dari sistem, bagaimana transaksinya ? Bagaimana sistem mengatasi error yang terjadi ? Bagaimana keamanan terhadap sistem yang ada kita buat ?Dan sebagainya dapat dijawab dengan UML.

UML diaplikasikan untuk maksud tertentu, biasanya diantara lain untuk :

1. Merancang Perangkat Lunak.
2. Sarana komunikasi antara perangkat lunak dengan bisnis
3. Menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem.
4. Mendokumentasikan sistem yang ada, proses-proses dan organisasinya.

UML telah diaplikasikan dalam investasi perbankan, lembaga kesehatan, departemen pertahanan, sistem terdistribusi, sistem pendukung alat kerja, retail, sales dan supplier.

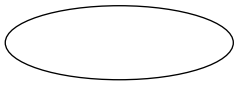
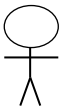
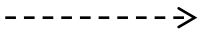
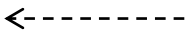
Blok pembangunan utama UML adalah diagram. Beberapa diagram ada yang di rinci (jenis timing diagram) dan lainnya ada yang bersifat umum (misalnya diagram kelas). Para pengembang sistem berorentasikan objek menggunakan bahasa model untuk menggambarkan, membangun dan mendokumentasikan sistem yang mereka rancang. UML memungkinkan para anggota team untuk bekerja sama dalam mengaplikasikan beragam sistem. Intinya, UML merupakan alat komunikasi yang konsisten dalam mensuport para pengembangan sistem saat ini. Sebagai perancang sistem mau tidak mau pasti menjumpai UML, baik kita sendiri yang membuat sekedar membaca diagram UML buatan orang lain (Prabowo Pudjo Widodo dan Herlawati, 2011).

Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan *UML* adalah sebagai berikut:

1.*Use case* Diagram

Use case diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram dapat dilihat pada tabel II.4 dibawah ini:

Tabel II.4. Simbol *Use Case*

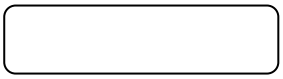
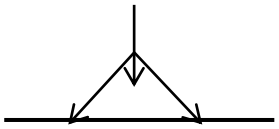
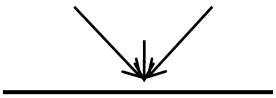
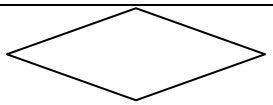
Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *use case*, tetapi tidak memiliki control terhadap *use case*.
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	*Include,* merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	*Extend,* merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

(Sumber: Gellysa Urva dan Helmi Fauzi Siregar; 2015)

2. Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram* dapat dilihat pada tabel II.5 dibawah ini:

Tabel II.5. Simbol *Activity Diagram*

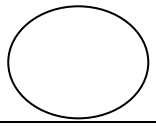
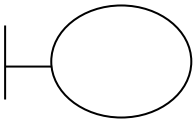
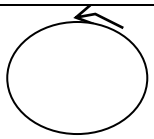
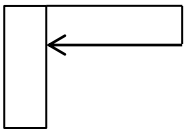
Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true*, *false*.
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

(Sumber : Gellysa Urva dan Helmi Fauzi Siregar, 2015 :94)

3. Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram* dapat dilihat pada tabel II.6 dibawah ini :

Tabel II.6. Simbol *Sequence Diagram*

Gambar	Keterangan
	*EntityClass*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*Boundary Class*, berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form* cetak.
	*Control class*, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
	*Message*, simbol mengirim pesan antar *class*.
	*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation*, *activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

(Sumber : Gellysa Urva dan Helmi Fauzi Siregar, 2015 :95)

5.*Class Diagram* (Diagram Kelas)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem.

Class diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*,

Generalization dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*), *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut.

Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti yang dapat dilihat pada tabel II.7 dibawah ini:

Tabel II.7. *Multiplicity Class Diagram*

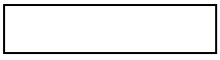
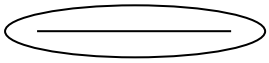
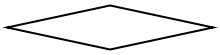
Multiplicity	**Penjelasan**
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

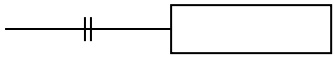
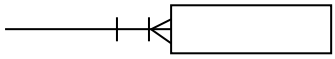
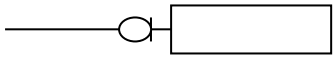
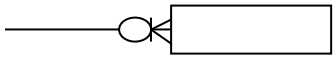
(Sumber : Gellysa Urva dan Helmi Fauzi Siregar, 2015: 95)

II.5. *Entity Relationship Diagram*

Entity Relationship Diagram (ERD) adalah bagian yang menunjukkan hubungan antara entity yang ada dalam sistem.Simbol-simbol yang digunakan dapat dilihat dari tabel II.8. (Yuhendra, M.T, Dr. Eng dan Riza Eko Yulianto, 2015: 70).

Tabel II.8. Simbol Yang Digunakan Pada *Entity Relationship Diagram (ERD)*

SIMBOL	**KETERANGAN**
	*Entity*
	Atribut Dan *Entity*
	Atribut Dan *Entity* Dengan *Key* (Kunci)
	Relasi Atau Aktifitas Antar *Entity*

	Hubungan Satu Dan Pasti
	Hubungan Banyak Dan Pasti
	Hubungan Satu Tapi Tidak Pasti
	Hubungan Banyak Tapi Tidak Pasti

(Sumber : Yuhendra, M.T, Dr. Eng dan Riza Eko Yulianto; 2015: 70)

II.6. Diagram-Diagram UML

Beberapa interatur menyebutkan bahwa UML menyebutkan sembilan jenis diagram, yang lain menyebutkan delapan karena ada beberapa yang digabung, misalnya diagram komunikasi, diagram urutan, dan diagram pewaktu digabung menjadi diagram interaksi. Namun demikian model-model ini dapat dikelompokan berdasarkan sifatnya yaitu statis atau dinamis. Jenis diagram itu antara lain :

1. Diagram Kelas Bersifat statis. Diagram ini memperlihatkan himpunan kelas-kelas, antar muka, kolaborasi, serta relasi-relasi diagram. Diagram ini umum dijumpai pada pemodelan sistem berorientasi objek. Meskipun bersifat statis, sering pula diagram kelas memuat kelas-kelas.
2. Diagram Paket (*Package Diagram*) Bersifat statis.Diagram ini memperlihatkan kumpulan kelas-kelas merupakan bagian dari diagram komponen.
3. Diagram *Use Case* Bersifat Statis. Diagram ini memperlihatkan himpunan *use-case* dan aktor-aktor (suatu jenis khusu dari kelas). Diagram ini

terutama sangat penting untuk mengorganisasi dan memodelkan perilaku suatu sistem yang dibutuhkan serta diharapkan pengguna.

4. Diagram Interaksi dan *Sequence*(Urutan). Bersifat dinamis. Diagram urutan adalah diagram interaksi yang menekankan pada pengiriman pesan dalam waktu tertentu.
5. Diagram Komunikasi (*Comunication Diagram*) bersifat dinamis. Diagram sebagai pengganti diagram kolaborasi *UML* yang menekankan organisasi *structural* dari objek-objek yang menerima serta mengirim pesan.
6. Diagram Statechart (*Statechart Diagram*) Bersifat Dinamis. Diagram status memperlihatkan keadaan-keadaan pada sistem, memuat status (*State*), transisi kejadian serta aktifitas. Diagram ini terutama penting untuk memperlihatkan sifat dinamis dari antarmuka (*interface*), kelas, kolaborasi, dan terutama penting pada pemodelan sistem-sistem yang reaktif.
7. Diagram Aktivitas (Activity Diagram) Bersifat Dinami. Diagram aktivitas adalah tipe khusus dari diagram status yang memperlihatkan aliran dari suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi-fungsi suatu sistem dan member tekanan pada aliran kendali antar objek.
8. Diagram Komponen (Component Diagram) Bersifat Statis. Diagram komponen ini memperlihatkan organisasi serta keberantungan sistem/perangkat lunak pada komponen-komponen yang telah ada sebelumnya. Diagram ini berhubungan diagram kelas dimana komponen

dipetakan dalam satu atau lebih kelas-kelas. Antarmuka serta kolaborasi-kolaborasi.

9. Diagram Deployment (*Deployment Diagram*) Bersifat Statis. Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan (run time). Memuat simpul-simpul berserta komponen-komponen yang ada dialamatnya. Diagram Deployment berhubungan erat dengan diagram komponen dimana diagram ini memuat satu atau lebih komponen-komponen. Diagram ini sangat berguna saat memuat satu atau lebih komponen-komponen. Diagram ini sangat berguna saat aplikasi kita berlaku sebagai aplikasi yang dijalankan pada banyak mesin (*distributed computering*).Kesembilan diagram ini tidak mutlak digunakan dalam pengembangan perangkat lunak, semuanya dibuat sesuai dengan kebutuhan. Pada *UML* dimungkinkan kita menggunakan diagram-diagram lainnya misalnya *Data Flow Diagram, Entity Relationship Diagram* dan sebagainya (Prabowo Pudjo Widodo dan Herlawati, 2011).

    1. Diagram *Use Case* (*Use Case Diagram*)

       Use Case menggambarkan external view dari sistem yang akan kita buat modelnya. Menurut (Pooley,2005) mengatakan bahwa model *use case* dapat dijabarkan dalam diagram, tetapi yang perlu diingat, diagram tidak indentik dengan model karena model lebih luas dari diagram. Komponen pembentuk diagram *use case*adalah :

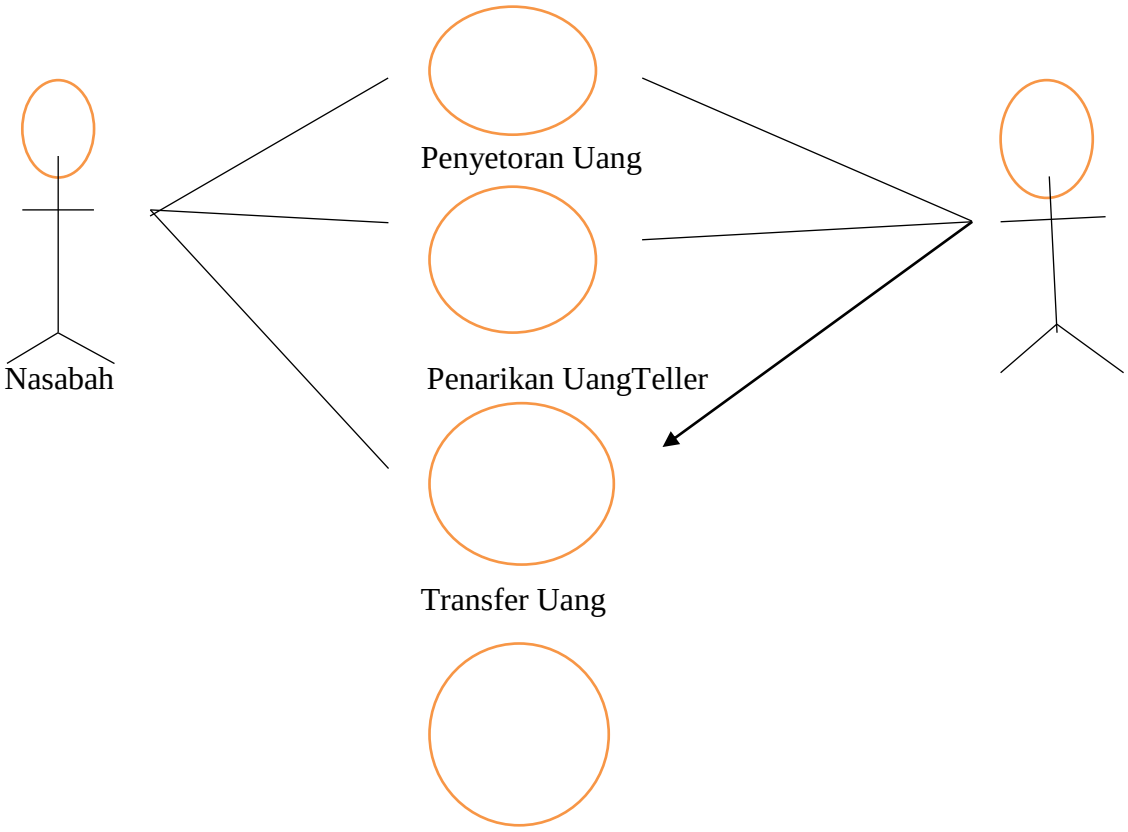
       a. Aktor (*actor*), menggambarkan pihak-pihak yang berperan dalam sistem.

b. *Use Case*, aktivitas / sarana yang disiapkan oleh bisnis/sistem.

c. Hubungan (*Link*), aktor mana saja yang terlibat dalam *use case* ini.

Gambar di bawah ini merupakan salah satu contoh bentuk diagram *use case*

(Prabowo Pudjo Widodo dan Herlawati, 2011).



Gambar II.1. Diagram *Use Case*
Sumber : (Prabowo Pudjo Widodo dan Herlawati, 2011)

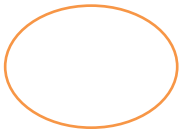
2. Aktor

Menurut (Chonoles, 2003) menyarankan sebelum membuat use case dan menentukanAktornya, agar mengidentifikasi siapa saja pihak yang terlibat dalam sistem kita. Pihak yang terlibat dalam sistem kita.Pihak yang terlibat biasanya dinamakan stakeholder.

Gambar II.2.Aktor
Sumber : (Prabowo Pudjo Widodo dan Herlawati, 2011)

3. *Use Case*

Menurut (Pilone, 2005) use case menggambarkan fungsi-fungsi tertentu dalam suatu sistem berupa komponen kejadian atau kelas. Sedangkan menurut (Whitten, 2004) mengartikanuse case sebagai urutan langkah-langkah yang secara tindakan saling terkait(skenario) baik terotomatisasi maupun secara manual, untuk tujuan melengkapi satu tugas bisnis tunggal. Use case digambarkan dalam bentuk *ellips/oval.*



Gambar II.3. Simbol Use Case
Sumber : (Prabowo Pudjo Widodo dan Herlawati, 2011)

Use case sangat menentukan karakteristik sistem yang kita buat, oleh karena itu (Chonoles, 2003) menawarkan cara untuk menghasilkan use case yang baik yakni :

a. Pilihlah Nama Yang Baik

   Use Case adalah sebuah *behaviour*(perilaku), jadi seharusnya dalam frase kata kerja. Untuk membuat namanya lebih detail tambahkan kata benda mengindikasikan dampak aksinya terhadap suatu kelas objek. Oleh karena itu diagram *use case* seharusnya berhubungan dengan diagram kelas.

b. Ilustrasikan Perilaku Dengan Lengkap.

   Use Case dimulai dari inisiasi oleh aktor primerdan berakhir pad aktor dan menghasilkan tujuan. Jangan membuat *use case* kecuali anda mengetahui tujannya. Sebagai contoh memilih tempat tidur (King Size, Queen Size, atau dobel) saat tamu memesan tidak dapat dijadikan use case karena merupakan bagian dari use case pemesanan kamar dan tidak dapat berdiri sendiri(tidak mungkin tamu memesan kamar tidur jenis king tapi tidak memesan kamar hotel).

c. Identifikasi Perilaku Dengan Lengkap.

   Untuk mencapai tujuan dan menghasilkan nilai tertentu dari aktor, use case harus lengkap. Ketika memberi nama pada use case, pilihlah frasa *reserve a room*(pemesanan kamar) dan jangan *reserving* a room(memesan kamar) karena memesan menggambarkan perilaku yang belum selesai.

d. Menyediakan *Use Case Lawan (Inverse)*

Kita biasanya membutuhkan use case yang membatalkan tujuan, misalnya pada use case pemesanan kamar, dibutuhkan pula use case pembatalan pesanan kamar.

e. Batasi Use Case Hingga Satu Perilaku Saja

Kadang kita cenderung membuat use case yang lebih dari satu tujan aktivitas. Guna menghindari keracunan, jagalah use case kita hanya fokus pada satu hal. Misalnya, penggunaan *use case check in* dan *check out* dalam satu menghasilkan ketidakfokusan, karena memiliki dua perilaku yang berbeda.

4. Diagram Kelas (Class Diagram)

Diagram kelas adalah inti dari proses pemodelan objek. Baik *forward engineering* maupun *reverse engineering* memanfaatkan diagram ini *forward enggineering* adalah proses perubahan model menjadi kode program sedangkan reverse engineering sebaliknya merubah kode program menjadi model (Prabowo Pudji Widodo dan Herlawati, 2011).

II.7. Mengenal Visual Basic

Visual Basic mengacu pada dua hal, yang pertama *Visual* dan yang kedua *Basic*. Kata "*Visual*" menunjukkan cara yang digunakan untuk membuat *Graphical User Interface* (GUI). Dengan cara ini progammer tidak lagi menuliskan instruksi pemrograman dalam kode-kode baris, tetapi secara mudah

*progammer*dapat melakukan "*drag and drop*" objek-objek yang akan digunakan. Kata "*Basic*" merupakan bagian bahasa BASIC (*Beginners All Purpose Symbolic Instruction Code*), yaitu sebuah bahasa pemrograman yang dalam sejarahnya sudah banyak digunakan oleh para progammeruntuk menyusun aplikasi.

Visual Basic merupakan salah satu bahasa pemrograman yang handal dalam lingkungan *Windows*. *Visual Basic* telah merajai pasar pembuatan *software*/ perangkat lunak sampai beberapa dekade tanpa ada yang menyaingi.*Visual Basic* 2010 merupakan teknologi terbaru yang masuk ke dalam *Visual Studio* bersama dengan C#, C++, dan yang lainnya.

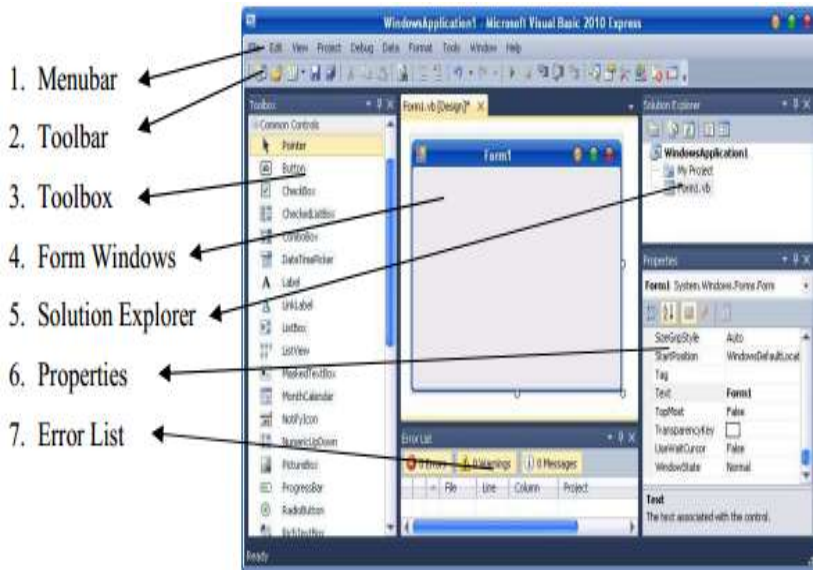
Visual Basic adalah salah satu *developmenttools* untuk membangun aplikasi dalam lingkungan *Windows*. *Visual Basic* menggunakan pendekatan *visual* untuk merancang *user interface* dalam bentuk form, sedangkan untuk *coding*-nya menggunakan dialek bahasa BASIC yang cenderung mudah dipelajari. Pada pemrograman *visual*, pengembangan aplikasi dimulai dengan pembentukan *user interface*, kemudian mengatur properties dari objek-objek yang digunakan dalam *user interface*, dan baru dilakukan penulisan kode program untuk menangani kejadian-kejadian (*event*).

II.8. *Microsft Visual Basic* 2010

Microsoft Visual Basic 2010 adalah salah satu komponen *Microsoft Visual Studio* 2010.*Software* ini diluncurkan *Microsoft* pada tanggal 12 April 2010 dengan nama kode Dev10 dan menggunakan .Net Framework 4.0. *Integrated*

Development Environment (IDE) pada *Visual studio* 2010 telah didesain ulang sehingga lebih enak dipandang dan digunakan *programmer*.

Untuk dapat menggunakan fasilitas dalam *Microsoft Visual Basic* 2010 dengan baik dan benar, maka diperlukan penguasaan tentang IDE (*Integratred Development Environment*) atau lingkungan kerja *Microsoft Visual Basic* 2010 itu sendiri. Tampilan fasilitas-fasilitas atau IDE *Microsoft Visual Basic* 2010 berisi komponen-komponen seperti terlihat dalam gambar berikut:



Gambar II.4*Interface Microsoft Visual Basic* 2010

Secara umum, IDE pada *Microsoft Visual Basic* 2010 terbagi menjadi 7 kompunen Besar, yaitu *Menu bar*, *Toolbar*, *Toolbox*, *Form Windows*, *Solution Explorer*, *Properties*, dan *Error List.*(Tim Penelitian dan Pengembangan Wahana Komputer; 2012 : 62)

II.9. SQL Server (*Structured Query Language*)

SQL (*Structured Query Language*) adalah bahasa yang digunakan untuk mengakses basis data yang tergolong relasional. Standar SQL mula-mula didefinisikan oleh ISO (*International Standards Organization*) dan ANSI (*The American National Standards Institute*), yang dikenal dengan sebutan SQL86.Standar terakhir berupa SQL99. SQL tidak terbatas hanya untuk mengambil data(*query*), tetapi dapat juga dipakai untuk menciptakan tabel, menghapus tabel, menambahkan data ke tabel, menghapus data di tabel, mengganti data di tabel, dan berbagai operasi yang lain (Abdul Kadir ; 2014 : 242).

Tabel II.9. Daftar sej*UML*ah pernyataan SQL

Pernyataan	Keterangan
SELECT	Untuk mengambil data
INSERT	Untuk menambahkan data
UPDATE	Untuk mengganti data
DELETE	Untuk menghapus data
CREATE TABLE	Untuk menciptakan tabel
DROP TABLE	Untuk menghapus tabel
GRANT	Untuk mengatur wewenang pemakai

Sumber: Abdul Kadir, 2014 : 242