BAB II

TINJAUAN PUSTAKA

II.1. Keputusan

Keputusan merupakan kegiatan memilih suatu strategi atau tindakan dalam pemecahan masalah tersebut. Tindakan memilih strategi atau aksi yang di yakini akan memberikan solusi terbaik atas sesuatu itu disebut pengambilan keputusan. Tujuan dari keputusan adalah untuk mencapai target atau aksi tertentu yang harus dilakukan (kusrini, 2007:6).

Kireteria atau cirri-ciri keputusan adalah:

1. Banyak pilihan/alternatif
2. Ada kendala atau syarat
3. Mengikuti suatu pola/model tingkah laku, baik yang terstruktur maupun tidak terstruktur.
4. Banyak input/variable
5. Ada faktor resiko
6. Dibutuhkan kecepatan, ketetapan, dan keakuratan.

Kondisi Pengambilan Keputusan

Ada beberapa keadaan yang mungkin dialami oleh pengambil keputusan ketika mengambil keputusan, yaitu:

1. Pengambil keputusan dalam kepastian, semua alternatif diketahui secara pasti.
2. Pengambilan keputusan dalam berbagai tingkat resiko yang dipilih.

3. Pengambilan keputusan dalam kondisi ketidak pastian, ada alternatif yang tidak diketahui secara jelas.

Tentu saja, pengambilan keputusan akan menjadi mudah jika dilakuakn dengan suatu kepastian.

II.2. Sistem Pendukung Keputusan

Sistem pendukung keputusan merupakan sistem informasi interaktif yang menyediakan informasi, pemodelan, dan pemanipulasian data. Sistem ini digunakan untuk mengambil keputusan dalam situasi yang semiterstruktur dan situasi yang tidak terstruktur, dimana tak seorang pun tahu secara pribadi bagaimana keputusan seharusnya dibuat (Kusrini, 2007:15).

Keputusan yang diambil untuk menyelesaikan suatu masalah dilihat dari keterstrukturannya yang bisa dibagi menjadi:

1. Keputusan terstruktur *(structure*
2. *d decision)*

   Keputusan terstruktur adalah keputusan yang dilakukan secara berulang-ulang dan bersifat rutin. Prosedur pengambilan keputusan sangatlah jelas. Keputusan tersebut terutama dilakukan pada manajemen tingakat bawah. Misalnya, keputusan pemesanan barang dan keputusan penagihan piutang.

3. Keputusan semiterstruktur *(semistructured decision)*

   Keputusan semiterstruktur adalah keputusan yang dimiliki dua sifat. Sebagai keputusan bisa ditangani oleh komputer dan yang lain tetap harus dilakukan oleh pengambil keputusan. Prosedur dalam pengambilan keputusan tersebut

secara garis besar sudah ada, tetapi ada beberapa hal yang masih memerlukan kebijakan dari pengambil keputusan. Biasanya, keputusan semacam ini diambil oleh manajer level menengah dalam suatu organisasi. Contoh keputusan jenis ini adalah pengevaluasian kredit, penjadwalan produksi, dan pengendalian sediaan.

4. Keputusan tak terstruktur *(unstructured decision)*

   Keputusan tak terstruktur adalah keputusan yang penanganannya rumit karena tidak terjadi berulang-ulang atau tidak selalu terjadi. Keputusan tersebut menuntut pengalaman dan berbagai sumber yang bersifat eksternal. Keputusan tersebut umumnya terjadi pada manajemen tingkat atas. Contohnya adlah keputusan untuk pengembangan teknologi baru, keputusan unutk bergabung dengan perusahaan lain, dan perekrutan eksekutif. (Kusrini, 2007:19).

Tujuan dari DSS adalah:

a. Membantu manajer dalam pengambilan keputusan atas masalah semi terstruktur.
b. Memberikan dukungan atas pertimbangan manajer dan bukannya dimaksudkan untuk menggantikan fungsi manajer.
c. Meningkatkan efektifitas keputusan yang diambil manajer lebih dari pada perbaikan efesiensinya.
d. Kecepatan komputasi. Komputer memungkinkan para pengambilan keputusan untuk melakukan banyak komputasi secara cepat dengan biaya yang rendah.

e. Peningkatan produktivitas.

f. Dukungan kualitas.

g. Berdaya saing.

h. Mengatasi keterbatasan kognitif dalam pemrosesan dan penyimpanan (Kusrini, 2007:16).

II.3. *System Additive Weighting* (SAW)

Metode SAW merupakan metode yang juga dikenal dengan metode penjumlahan berbobot. Konsep dasar metode SAW adalah mencari penjumlahan terbobot dari rating kinerja pada setiap alternatif pada semua atribut (Fishburn, 1967) (MacCrimmon, 1968).

Metode SAW membutuhkan proses normalisasi matriks keputusan (X) ke suatu skala yang dapat diperbandingkan dengan semua rating alternatif yang ada (Kusumadewi, 2006 : 74).

Metode ini merupakan metode yang paling terkenal dan paling banyak digunakan dalam menghadapi situasi Multiple Attribute Decision Making (MADM). MADM itu sendiri merupakan suatu metode yang digunakan untuk mencari alternatif optimal dari sejumlah alternatif dengan kriteria tertentu.

$$r_{ij} = \begin{cases} \dfrac{X_{ij}}{\max_i X_{ij}} & \text{Jika j adalah atribut keuntungan (benefit)} \\ \min_i X_{ij} & \end{cases}$$

$\frac{}{}$ Jika j adalah atribut biaya (cost)

Keterangan :

Rij = nilai rating kinerja normalisasi

Xij = nilai atribut yang dimiliki dari setiap kriteria

Max xij = nilai terbesar dari setiap kriteria
Min xij = nilai terkecil dari setiap kireteria

Benefit = nilai terbesar adalah terbaik

Cost = nilai terkecil adalah terbaik.

Dimana rij adalah rating kinerja ternormalisasi dari alternatif Ai pada atribut Cj; i=1,2,...,m dan j=1,2,...,n. Nilai preferensi untuk setiap alternatif (Vi) diberikan sebagai.

$$Vi = \sum_{j=i}^{n} w_j r_{ij}$$

Vi = rangking untuk setiap alternatif

wj = nilai bobot dari setiap kriteria

rij = nilai rating kinerja ternormalisasi

Nilai Vi yang lebih besar mengindikasikan bahwa alternatif Ai lebih terpilih. Langkah dari metode SAW adalah:

1. Menentukan kireteria-kireteria yang akan dijadikan acuan dalam pengambilan keputusan.
2. Menentukan rating kecocokan setiap alternatif pada setiap kriteria.

3. Membuat keputusan berdasarkan kriteria, kemudian melakukan normalisasi matriks berdasarkan persamaan yang disesuaikan dengan jenis atribut.
4. Hasil akhir diperoleh dari proses perangkingan yaitu penjumlahan dari perkalian ternormalisasi dengan bobot sehingga diperoleh nila terbesar yang dipilih sebagai alternatif terbaik sebagai solusi (Destriyani Darmastuti; 3).

II.3.1. Kelebihan Metode SAW

Kelebihan metode SAW di bandingkan dengan model pengambilan keputusan yang lain terletak pada kemampuanya untuk melakukan penelitian secara lebih tepat karena didasarkan pada nilai kireteria dan bobot preferensi yang sudah ditentukan, selain itu SAW juga dapat menyeleksi alternatif terbaik dari sejumlah alternatif yang ada karena adanya proses perangkingan setelah menentukan nilai bobot untuk setiap atribut (Destriyana Darmastuti ; 3).

II.4. Entity Relationship Diagram (ERD)

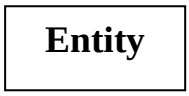
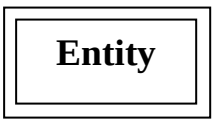
ERD adalah suatu model jaringan yang menggunakan susunan data yang disimpan dalam sistem secara abstrak. Jadi, jelaslah bahwa ERD ini berbeda dengan DFD yang merupakan suatu model jaringan fungsi yang akan dilaksanakan oleh sistem, sedangkan ERD merupakan model jaringan data yang menekankan pada struktur-struktur dan relationship data (Al-Bahra bin Ladja Muddin B ; 2004 : 123).

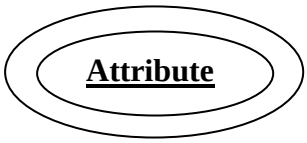
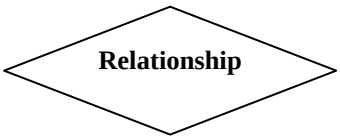
II.4.1 Notasi Diagram Relasi

Peter chan mengembangkan ERD pada tahun 1972. Kemudian, Charles Bachman dan James Martin menambahkan beberapa perbaikan dalam prinsip-prinsip dasar ERD (Janner Simarmata, 2007; 113).

Adapun simbol notasi diagram relasi dapat dilihat di tabel II.1 berikut:

Table II.1 Diagram Relasi

	Entity Suatu entity merupakan suatu objek atau konsep mengenai tempat yang anda inginkan untuk menyimpan informasi
	Weak Entity Suatu weak entity tergantung pada entitas lainya
	Attributes Attributes adalah sifat-sifat atau karakteristik dari suatu entitas.
	Key Attribute Suatu key attribute adalah unik (unique), dan memiliki karakteristik pembeda dari entitas. Sebagai contoh, nomor mahasiswa mungkin menjadi attribute key mahasiswa.

	Multivalued attribute Suatu multivalued attribute memiliki lebih dari satu nilai. Sebagai contoh, suatu entitas pegawai bias memiliki nilai pada berbagai keahlian.
	Derived attribute Suatu derived attribute didasarkan pada attribute lainnya. Sebagai contoh, gaji bulanan seseorang pegawai berdasarkan pada gaji bulanan karyawan lain yang berdasarkan pada gaji tahunan.
	Relationships Relationship mengilustrasikan bagaimana dua entitas berbagai informasi di dalam struktur basis data.

Sumber : (Janner Simarmata, 2007: 112)

II.5. Normalisai

Normalisasi adalah proses pengelompokan data ke dalam bentuk tabel atau relasi atau file untuk menyatakan entitas dan hubungan mereka sehingga terwujud satu bentuk database yang mudah di modifikasi (Al-Bahra bin Ladja Muddin B ; 2004 : 174).

II.5.1. Langkah-Langkah Pembentukan Normalisasi

1. Bentuk Tidak Normal (Unnormalized Form)

Bentuk ini merupakan kumpulan data yang akan di rekam, tidak ada keharusan mengikuti format tertentu, dapat saja data tidak lengkap atau terduplikasi. Data dikumpulkan apa adanya sesuai dengan mengimput.

Contoh yang belum normal bisa kita lihat di tabel II.2.

Table II.2 Tabel Yang Belum Normal

No_Pe Langgan	Nama	Nomor Pro-perty	Alamat property	Tgl_pin jam	Tgl_se lesai	Biaya	No_pe milik	Nama_pe milik
CR76	Badi	PG4	Jl.Aai / 07,jakarta	1-jul-93	31-aug-95	350	CO4O	Ewin
		PG16	Jl.hunzai/12, jakarta	1-sep-95	1-sep-96	450	CO93	Durki
CR56	Sirajuddin	PG4	Jl.Aai / 07,jakarta	1-sep-92	10-jun-93	350	C040	Ewin
		PG36	Jl.Azhar/49 ,jakarta	10-oct-93	1-des-94	375	CO93	Durki
		PG16	JL.Huzai/12, jakarta	1-jan-95	10-aug-95	450	CO93	Durki

Sumber : (Al-Bahra Bin Ladjamuddin, 2004: 183)

Dari Tabel II.2 di atas terdapat multivalue pada beberapa atributnya. Misalkan terdapat dua (2) nilai untuk No_Property yaitu PG4 dan PG16 untuk Nama Pelanggan (Nama) Badi. Untuk Mentranformasikan tabel yang belum ternormalisasi di atas menjadi tabel yang memenuhi kireteria 1NF adalah kita harus merubah seluruh atribute yang multi value menjadi atribute yang single value, dengan cara menghilangkan repeating group pada tabel di atas.

Repeating Group (elemen data berulang) adalah

(No_property, Alamat_property, Tgl_pinjaman, Tgl_selesai, Biaya, No_pemilik, Nama_pemilik) (Al-Bahra Bin Ladjamuddin, 2004: 184).

2. Bentuk Normal Ke Satu (1-NF)

   Pada tahap ini dilakukan penghilangan beberapa group elemen yang berulang agar menjadi satu harga tunggal yang berinterkasi di antara setiap baris pada suatu tabel, Syarat normal kesatu (1-NF) yaitu tidak ada set atribute yang berulang atau bernilai ganda.

Contoh bentuk normal ke satu (1NF) bisa kita lihat di tabel II.3

Table II.3 Tabel Bentuk Ke Satu (1NF)

No_Pe Langgan	Nama	Nomor Pro-perty	Alamat property	Tgl_pin jam	Tgl_se lesai	Biaya	No_pe milik	Nama_pe milik
CR76	Badi	PG4	Jl.Aai / 07,jakarta	1-jul-93	31-aug-95	350	CO4O	Ewin
CR76	Badi	PG16	Jl.hunzai/12, jakarta	1-sep-95	1-sep-96	450	CO93	Durki
CR56	Sirajuddin	PG4	Jl.Aai / 07,jakarta	1-sep-92	10-jun-93	350	C040	Ewin
CR56	Sirajuddin	PG36	Jl.Azhar/49 ,jakarta	10-oct-93	1-des-94	375	CO93	Durki
CR56	Sirajuddin	PG16	JL.Huzai/12, jakarta	1-jan-95	10-aug-95	450	CO93	Durki

Sumber : (Al-Bahra Bin Ladjamuddin, 2004: 185)

Kita dapat mengidentifikasi primary key untuk relasi pelanggan_biaya yang masih memiliki composite key (No_pelanggan, No_property). Pada kasus ini kita akan memperoleh primary key yang bersifat composite key.

Relasi Pelanggan_biaya dapat = (No_pelanggan, No_property, Nama, Alamat_property, Tgl_pinjaman, Tgl_selesai, Biaya, No_pemilik, Nama_pemilik) (Al-Bahra Bin Ladjamuddin, 2004: 183).

3. Bentuk Normal Ke Dua (2-NF)

   Bentuk normal kedua didasari atasa konsep full functional dependency (ketergantungan fungsional sepenuhnya). Syarat normal kedua (2-NF) yaitu bentuk telah memenuhi kireteria bentuk normal kesatu, dan attribute bukan kunci (non-key) haruslah memiliki ketergantungan fungsional sepenuhnya pada kunci utama/primary key Al-Bahra bin Ladja Muddin B ; 2004 : 187).

Contoh Tabel yang teleh memenuhi kiriteria normal ke 2 (2NF) bisa di lihat di ta bel II.4.

Table II.4 Tabel Bentuk Ke Dua (2NF)

No_pelanggan	Nama
CR76	Badi
CR56	Sirajuddin

(a) Relasi Pelanggan

No_Pelanggan	No_property	Tgl_pinjaman	Tgl_selesai
CR76	PG4	1-Jul-93	31-aug-95

CR76	PG16	1-sep-95	1-sep-96
CR56	PG4	1-SEP-92	10-JUN-93
CR56	PG36	10-0ct-93	1-dec-94
CR56	PG16	1-jan-95	10-aug-95

(b) Relasi Biaya

No_property	Alamat_property	Biaya	No_pemilik	Nama_pemilik
PG4	Jl.Aai/07, Jakarta	3530	C040	Ewin
PG16	Jl.Huzai/12, jakarta	450	C093	Durki
PG36	Jl.Suciana / 68, Bogor	375	C093	Durki

(c) Relasi Property_pemilik

Sumber : (Al-Bahra Bin Ladjamuddin, 2004: 190)

Agar seluruh atribute non-key (yang bukan primary key) dapat mimiliki ketergantungan sepenuhnya (fully functionally dependent) terhadap primary key. Ketiga buah relasi tersebut dapat di tulis dalam bentuk.

1. Relasi/tabel pelanggan dengan atribute: N0_pelanggan, Nama.
2. Relasi/tabel biaya dengan atribute: No_pelanggan, No_property, Tgl_pinjaman, Tgl_selesai.
3. Relasi/tabel Property_pemilik dengan atribute: No_property, alamat_property, biaya, No_pemilik, Nama_pemilik (Al-Bahra Bin Ladjamuddin, 2004: 189).

4. Bentuk Normal Ke Tiga (3-NF)

   Bentuk data telah memenuhi kriteria bentuk normal kedua, atribute bukan kunci (non-key) harus tidak memiliki ketergantungan transistif, dengan kata lain suatu atribute bukan kunci (non-key) tidak boleh memiliki ketergantungan fungsional (fungsioanal depedency) terhadap atribute bukan kunci lainya (Al-Bahra Bin Ladjamuddin, 2004: 191)

Contoh Tabel yang teleh memenuhi kiriteria normal ke 2 (2NF) bisa di lihat di ta bel III.5.

Table II.5 Tabel Bentuk Ke Tiga (3NF)

No_pelanggan	Nama
CR76	Badi
CR56	Sirajuddin

(a) Relasi Customer

No_Pelanggan	No_property	Tgl_pinjaman	Tgl_selesai
CR76	PG4	1-Jul-93	31-aug-95
CR76	PG16	1-sep-95	1-sep-96
CR56	PG4	1-SEP-92	10-JUN-93
CR56	PG36	10-0ct-93	1-dec-94
CR56	PG16	1-jan-95	10-aug-95

(b) Relasi Rental

No_property	Alamat_property	Biaya	No_pemilik

PG4	Jl.Aai/07, jakarta	3530	C040
PG16	Jl.Huzai/12, jakarta	450	C093
PG36	Jl.Suciana / 68, Bogor	375	C093

(c) Relasi Property

No_pemilik	Nama_pemilik
C040	Erwin
C093	Durki
C093	Durki

(d) Relasi Owner

Sumber : (Al-Bahra Bin Ladjamuddin, 2004: 195.

II.6. Visual Studio 2010

Visual Studio 2010 Merupakan sebuah *Integrated Devlopment Environtment* (IDE) atau Lingkungan kerja yang digunakan unruk membangun aplikasi **.**NET yang dapat digunakan untuk beberapa bahasa pmograman, seperti visual basic (VB), C#, Visual C++, J#, F# dan lain-lain. *Visual Studio Profesional* 2010 menyediakan berbagai tool yang lengkap bagi para pengembang untuk membangun aplikasi yang berjalan di .*Net Framework*. Berbagai tool, antara lain tool Toolbox yang berisi komponen visual, selain itu ada jendela wizard yang membantu untuk melakukan pemograman dengan sangat mudah (Wahana Komputer, 2012; 7).

II.6.I Eksekusi Kode

Bahasa program adalah sebuah bahasa yang digunakan oleh manusia untuk berhubung/berinteraksi dengan komputer, sehingga komputer mengerti apa yang kita maksudkan. Bahasa yang digunakan mesin adalah kode mesin dan hanya berupa 0 dan 1, semua saling berhubungan secara elektris (Wahana Komputer, 2012

II.7. SQL Server 2008

SQL Server 2008 adalah sebuah RDBMS (*Relational Database Managemennt System*) yang di develop oleh Microsoft, yang digunakan untuk menyimpan dan mengolah data. Pada SQL Server 2008 kita bisa melakukan pengambilan dan modifikasi data yang ada dengan cepat dan efisien.Pada SQL Server 2008 kita bisa membuat object-object yang sering digunakan pada aplikasi bisnis, seperti membuat database, table, function, stored procedure, trigger dan view. Selain object, kita juga menjalankan perintah SQL (*Structured Query Language*) untuk mengambil data.

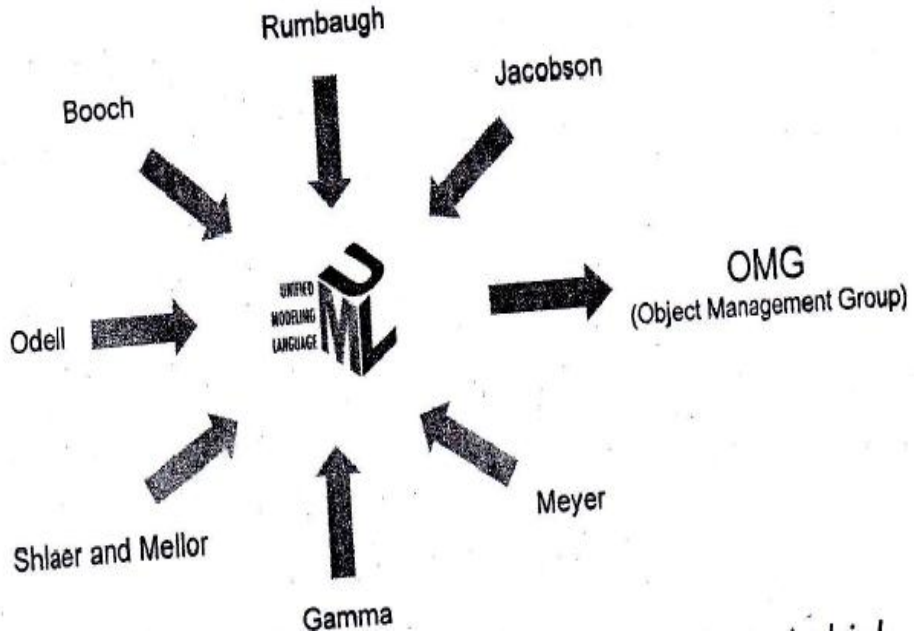
II.8. UML (Unified Modeling Language)

Unified Modeling Language adalah sebuah “bahasa” yang telah menjadi setandar dalam industry untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem (Yuni Sugiarti;2013:34).

Dengan menggunakan UML kita dapat membuat untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi, dan jaringan apapun, serta ditulis dalam bahasa pemograman

apapun. Tetapi karena UML juga menggunaka clas 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 procedural dalam VB atau C.

Seperti bahasa lainnya, UML mendefenisikan notasi dan syntax/semantic. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML syntax mendefenisikan 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 Enginering).



Gambar II.1. metodelogi pemodelan berorientasi objek

Sumber : (Yuni Sugiarti;2013:35)

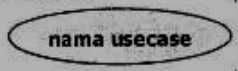
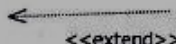
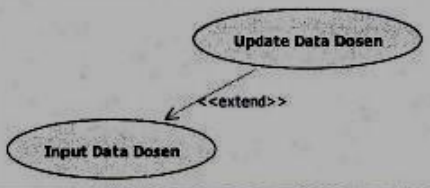
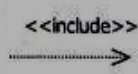
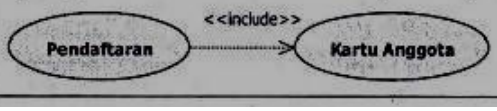
II.8.1. Use Case Digram

Dalam membuat sebuah sistem, langkah awal yang perlu dilakukan adalah menentukan kebutuhan.Terdapat dua kebutuhan yaitu kebutuhan fungsional dan nonfungsional. Kebutuhan fungsional adalah kebutuhan pengguna dan stakeholder sehari-hari yang akan dimiliki oleh sistem, dimana kebutuhan ini akan digunakan oleh pengguna atau stakeholder. Sedangkan kebuthan nonfungsional adalah kebutuhan yang memperhatikan hal-hal berikut yaitu performansi, kemudahan dalam menggunakan sistem, kehandalan sistem, keamanan sistem, keuangan, legalitas, operasional.

Kebutuhan fungsional akan digambarkan melalui sebuah gambar yang dinamakan diagram use case. Use Case Diagram atau diagram use case merupakan pemodelan untuk menggambarkan kelakuan (behavior) sistem yang akan dibuat. Diagram Use Case mendeskripsikan sebuah interaksi antara satu atau lebih actor dengan sistem yang dibuat. Dengan pengertian yang cepat, diagram use case digunakan untuk mengetahui fungsi apa saja yang ada dalam sebuah sistem dan apa saja yang berhak menggunakan fungsi-fungsi tersebut. Terdapat beberapa simbol dalam menggambarkan use case, yaitu use case, actor, relasi.

Adapun simbol use case diagram dapat dilihat di table II.6. berikut:

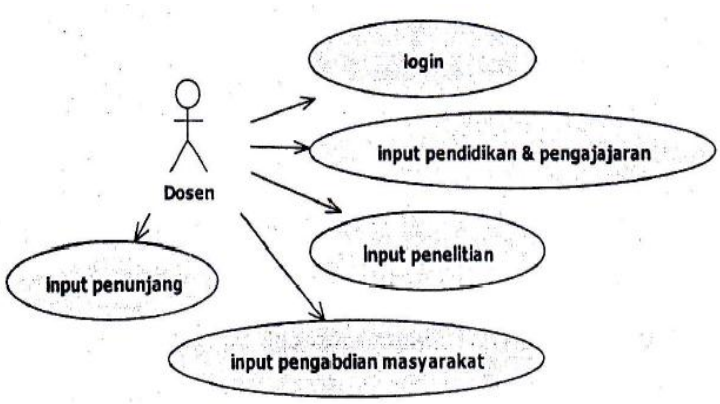
Tabel II.6. Simbol Use Case Diagram

Simbol	Deskripsi
Use case 	fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awalfrase namause case
Aktor  nama actor	orang, proses, atau sistem lain yangberinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yangakan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang;biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.
Asosiasi / association	komunikasi antara aktor danuse caseyangberpartisipasi pada use case atauuse casememiliki interaksi dengan aktor
Extend 	relasiuse case tambahan ke sebuah use case dimana use caseyang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan itu; mirip dengan prinsip inheritance pada pemrograman berorientasi objek; biasanya use casetambahan memiliki nama depan yang sama dengan use case yang ditambahkan, arah panah menunjuk pada use case yang dituju.contoh : 
Include 	relasiuse case tambahan ke sebuahuse casedimanause case yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankanusecaseini. Ada dua sudut pandang yang cukup besar mengenai include diuse case, include berartiuse case yang ditambahkan akan selalu dipanggil saat use case tambahan dijalankan, contoh : 

Sumber : (Yuni Sugiarti;2013:42)

Use Case Diagram adalah sebuah diagram yang menjelaskan apa yang harus dilakukan oleh sistem pada level konseptual sehingga kita akan memahami apakah keputusan yang diambil oleh sistem adalah benar atau tidak.

Adapun contoh use case adalah ditujukan oleh gambar II.3 sebagai berikut:



Gambar II.2. contoh Use Case Diagram

Sumber : (Yuni Sugiarti;2013:45)

II.8.2. Class Diagram

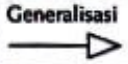
Diagram kelas atau class diagram menggambarkan struktur sistem dari segi pendefenisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

- Atribut merupakan variable-variabel yang dimiliki oleh suatu kelas.

- Atribut mendeskripsikan property dengan sebaris teks di dalam kotak tersebut.
- Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh satu kelas.

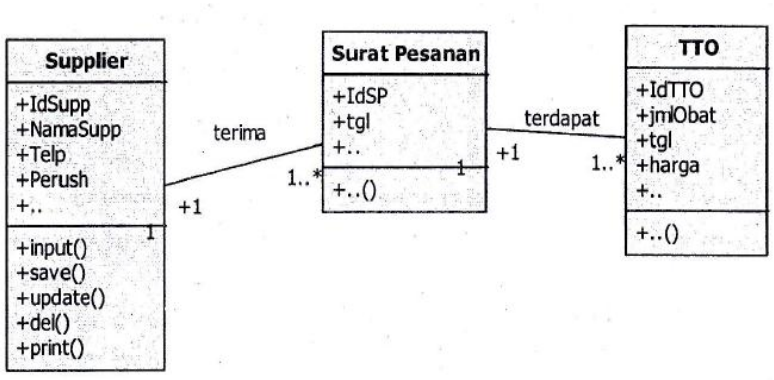
Diagram kelas mendeskripsikan jenis jenis objek dalam sistem dan berbagai hubungan statis yang terdapat diantara mereka.Diagram kelas juga menunjukan property dan operasi sebuah kelas dan batasan-batasanyang terdapat dalam hubungan-hubungan objek tersebut. Berikut adalah simbol-simbol yang ada pada diagram kelas dapat dilihat pada tabel II.7.

Tabel II.7. Simbol Class Diagram

Simbol	Deskripsi
Package 	Package merupakan sebuah bungkusan dari satu atau lebih kelas
Operasi 	Kelas pada struktur sistem
Antarmuka / interface 	sama dengan konsepinterface dalam pemrograman berorientasi objek
Asosiasi 	relasi antar kelas dengan maknaumum, 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 jugadisertai dengan multiplicity
Generalisasi 	relasi antar kelas dengan maknageneralisasi-spesialisasi (umumkhusus)
Kebergantungan/ defedency 	relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi	relasi antar kelas dengan maknasemua-bagian (whole-part)

Sumber : (Yuni Sugiarti;2013:59)

Adapun contoh Class Diagram dapat dilihat pada gambar II.3.berikut:



Gambar II.3. contoh Class Diagram

Sumber : (Yuni Sugiarti;2013:60)

II.8.3. Activity Diagram

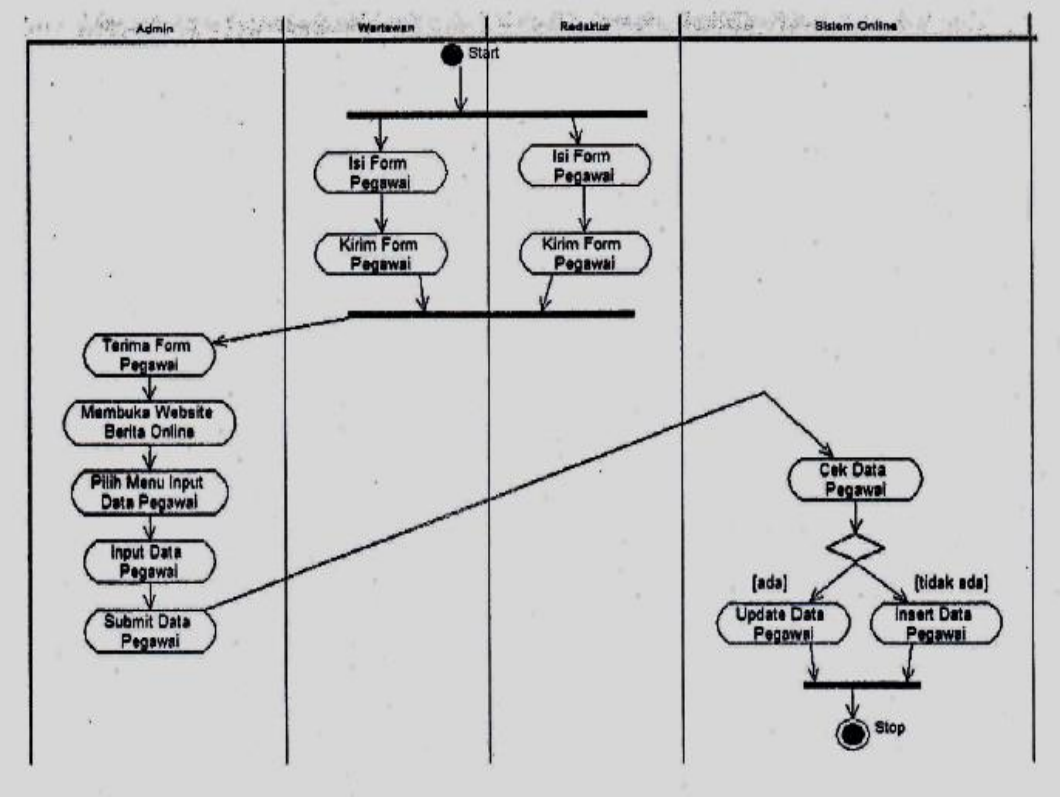
Diagram aktivitas atau activity diagram menggambarkan workflow (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu di perhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan actor, jadi aktivitas yang dapat dilakukan oleh sistem. Diagram aktivitas mendukung prilaku paralel.

Diagram aktivitas juga banyak digunakan untuk medefenisikan hal-hal berikut:

- Rancangan proses bisnis dimana setiap urutan aktivitas yang di gambarkan merupakan proses bisnis sistem yang di defenisikan.

- Urutan atau pengelompokan tampilan dari sistem/ user interface dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
- Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefenisikan kasus ujiannya.

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 meggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. Berikut adalah contoh activity diagram yang dapat dilihat pada gambar II.4.



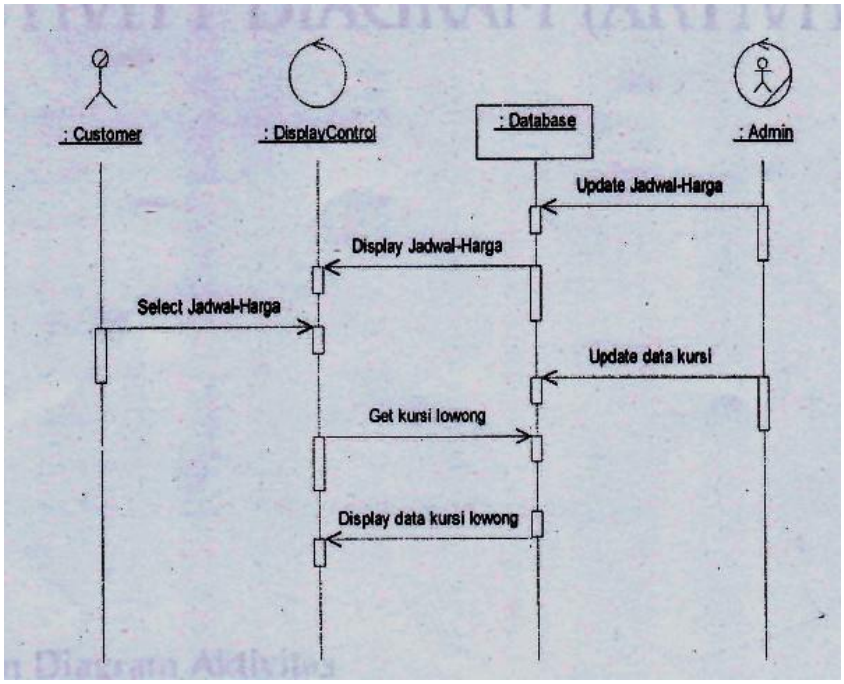
Gambar II.4. contoh Activity diagram

Sumber : (Yuni Sugiarti;2013:77)

II.8.4. Sequence Diagram

Sequence diagram menggambarkan kelakuan/prilaku objek pada use case dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan di terima antar objek. Oleh karena itu untuk menggambar sequence diagram maka harus di ketahui objek-objek yang terlibat dalam sebuah use case beserta metode-metode yang dimiliki kelas yang diintansiasi menjadi objek tertentu.

Banyaknya diagram sequence yang harus digambar adalah sebanyak pendefenisian use case yang memiliki proses sendiri atau yang penting semua use case yang telah di defenisikan interaksi jalannya pesan sudah dicakup pada diagram sequence sehingga semakin banyak use case yang di defenisikan maka diagram sequence yang harus di buat juga semakin banyak. Berikut contoh Sequence Diagram yang dapat dilihat pada gambar II.5 berikut:



Gambar II.5. Contoh Sequnce Diargam

Sumber : (Yuni Sugiarti;2013:71)