BAB II

TINJAUAN PUSTAKA

II.1. Sistem

Sistem merupakan kumpulan elemen yang saling berkaitan yang bertanggung jawab memproses masukan (*input)* sehingga menghasilkan keluaran *(output).* Suatu sistem didalam suatu organisasi yang memperemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial, adan merupakan kegiatan strategi dari suatu organsisasi, serta menyediakan laporan – laporan yang diperlukan oleh pihak luar.

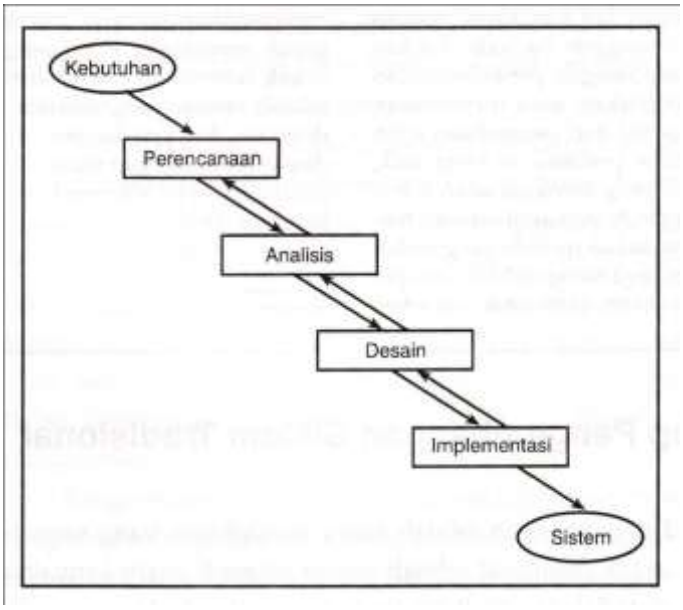
Berdasarkan dukungan kepada pemakainya, sistem informasi dibagi menjadi:

1. Sistem Pemrosesan Transaksi (*Transaction Processing System* atau TPS)
2. Sistem Informasi Manajemen (*Management Information System* atau MIS)
3. Sistem Otomasi Perkantoran (*Office Automation System/OAS)*
4. Sistem Pendukung Keputusan (*Decision Support System* atau DSS)
5. Sistem Informasi Eksekutif (*Executive Information System* atau EIS)
6. Sistem Pendukung Kelompok (*Group Support System* atau GSS)
7. Sistem Pendukung Cerdas (*Intelligent Support System* atau ISS).

   (Kusrini, M.Kom ; 2007 : 11)

Pengembangan adalah suatu pendekatan yang sangat rapi dan berurutan untuk membuat sebuah sistem menjadi suatu kenyataan. Diperlukan suatu

metodologi untuk menyediakan suatu struktur pengembangan sistem. Ada banyak Siklus Hidup Pengembangan Sistem (SDLC) "tradisional" untuk sistem informasi, termasuk DSS. Setiap perangkat rancang-bangun perangkat lunak dengan bantuan komputer (CASE) telah mengadopsi suatu variasi. Untuk berbagai hal sulit, masing-masing organisasi yang mengembangkan sebuah sistem dapat menciptakan variasi in-house agar sesuai dengan kebu spesifik mereka. Masing-masing metodologi menekankan langkah-langkah berbeda dalam cara yang berbeda. Akan tetapi, semua SDLCS yang d secara intuitif dan praktis harus mengikuti petunjuk dan proses tertentu. (Efraim Turban ; 2005 : 401-402).



Gambar II.1. SDLC

(Efraim Turban ; 2005 : 402)

II.1.1. Sistem Pendukung Keputusan

Sistem Pendukung Keputusan DSS (*Decision Support System*) merupakan sistem informasi interaktif yang menyediakan informasi, pemodelan, dan pemanipulasian data. Sistem itu digunakan untuk mem-bantu pengambilan keputusan dalam situasi yang semiterstruktur dan situasi yang tidak terstruktur, di mana tak seorang pun tahu secara pasti bagaimana keputusan seharusnya dibuat.
DSS biasanya dibangun untuk mendukung solusi atas suatu masalah atau untuk mengevaluasi suatu peluang. DSS yang seperti itu disebut aplikasi DSS. Aplikasi DSS digunakan dalam pengambilan keputusan. Aplikasi DSS menggunakan CBIS (*Computer Based Information Systems*) yang fleksibel, interaktif, dan dapat diadaptasi, yang dikembangkan untuk mendukung solusi atas masalah manajemen spesifik yang tidak terstruktur.

Aplikasi DSS menggunakan data, memberikan antarmuka pengguna yang mudah, dan dapat menggabungkan pemikiran pengambil keputusan. DSS lebih ditujukan untuk mendukung manajemen dalam melakukan pekerjaan yang bersifat analitis dalam situasi yang kurang terstruktur dan dengan kriteria yang kurang jelas. DSS tidak dimaksudkan untuk mengotomatisasikan pengambilan keputusan, tetapi memberikan perangkat interaktif yang memungkinkan pengambil keputusan untuk melakukan berbagai analisis menggunakan model-model yang tersedia.

Tujuan dari DSS adalah:

1. Membantu manajer dalam pengambilan keputusan atas masalah semi-terstruktur.

2. Memberikan dukungan atas pertimbangan manajer dan bukannya dimaksudkan untuk menggantikan fungsi manajer.
3. Meningkatkan efektivitas keputusan yang diambil manajer lebih dari-pada perbaikan efisiensinya.
4. Kecepatan komputasi. Komputer memungkinkan para pengambil ke-putusan untuk melakukan banyak komputasi secara cepat dengan biaya yang rendah.
5. Peningkatan produktivitas. Membangun satu kelompok pengambil keputusan, terutama para pakar, bisa sangat mahal. Pendukung terkomputerisasi bisa mengurangi ukuran kelompok dan memungkinkan para anggotanya untuk berada di berbagai lokasi yang berbeda-beda (menghemat biaya perjalanan). Selain itu, produktivitas staf pen-dukung (misalnya analis keuangan dan hukum) bisa ditingkatkan. Produktivitas juga bisa ditingkatkan menggunakan peralatan optimalisasi yang menentukan cara terbaik untuk menjalankan sebuah bisnis.
6. Dukungan kualitas. Komputer bisa meningkatkan kualitas keputusan yang dibuat. Sebagai contoh, semakin banyak data yang diakses, makin banyak juga alternatif yang bisa dievaluasi. Analisis resiko bisa dilakukan dengan cepat dan pandangan dari para pakar (beberapa dari mereka berada dilokasi yang jauh) bisa dikumpulkan dengan cepatdan dengan biaya yang lebih rendah. Keahlian bahkan bisa diambil langsung dari sebuah sistem komputer melalui metode kecerdasan

tiruan. Dengan komputer, para pengambil keputusan bisa melakukan simulasi yang kompleks, memeriksa banyak skenario yang memungkinkan, dan menilai berbagai pengaruh secara cepat dan ekonomis. Semua kapabilitas tersebut mengarah kepada keputusan yang lebih baik.

7. Berdaya saing. Manajemen dan pemberdayaan sumber daya perusahaan. Tekanan persaingan menyebabkan tugas pengambilan keputusan menjadi sulit, persaingan didasarkan tidak hanya pada harga, tetapi juga pada kualitas, kecepatan, kustomasi produk, dan dukungan pelanggan. Organisasi harus mampu secara sering dan cepat mengubah mode operasi, merekayasa ulang proses dan struktur, memberdayakan karyawan, serta berinovasi. Tekhnologi pengambilan keputusan bisa menciptakan pemberdayaan yang signifikan dengan cara memperbolehkan seseorang untuk membuat keputusan yang baik secara cepat, bahkan jika mereka memiliki pengetahuan yang kurang.
8. Mengatasi keterbatasan kognitif dalam pemrosesan dan penyimpanan (Kusrini : 2007 : 15-17)

II.2. Metode Fuzzy

Konsep Logika *Fuzzy* dicetuskan oleh Loth Zaden, seorang profesor University of California di Berkeley, dan dipresentasikan bukan sebagai metodologi kontrol, namun sebagai suatu cara pemrosesan data yang memperbolehkan anggota himpunan parsial daripada anggota himpunan kosong

atau non - anggota. Pendekatan ini pada teori himpunan tidak diaplikasikan untuk mengontrol sistem sampai tahun 70-an karena kurangnya kemampuan komputer mini pada saat itu. Profesor Zadeh beralasan bahwa masyarakat tidak butuh ketepatan, input informasi numeris, dan mereka belum sanggup dengan kontrol adaptif yang tinggi. Jika kembalian dari kontroler dapat diprogram untuk menerima noisy, input yang tidak teliti, mereka akan lebih efektif dan lebih mudah diimplementasikan.

II.2.1 Himpunan Fuzzy

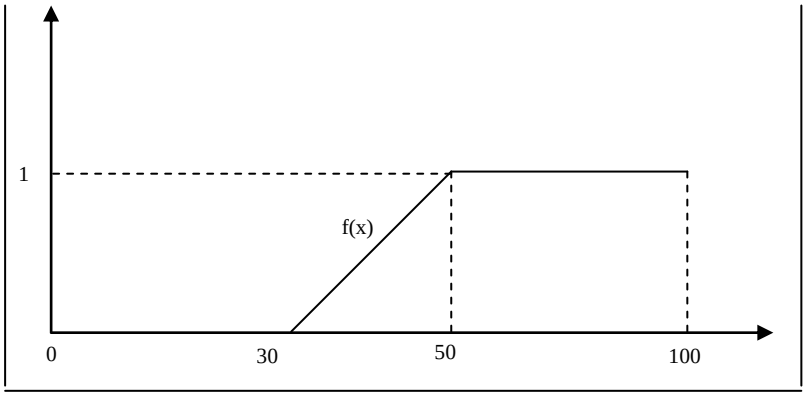
Sebuah Himpunan *fuzzy* dari semesta U dikelompokkan oleh fungsi keanggotaan $\mu\Lambda(x)$ yang berada pada nilai antara [0,1] (wang, 1997). Fungsi kenggotaan dari himpunan klasik hanya memiliki 2 nilai yaitu 0 dan 1, sedangkan fungsi keanggotaan himpunan fuzzy merupakan fungsi kontinu dengan *range* [0,1].

II.2.2 Fungsi Keanggotaan

Fungsi keanggotaan adalah sebuah representasi grafis dari besarnya partisipasi masing – masing input. Fungsi keanggotaan dihubungkan dengan pembobotan masing – masing input yang diproses, definisi pencocokan fungsi antar – input dan penentuan *respons* keluaran (Kaehler).

Sebagai contoh dalam menentukan fungsi keanggotaan, diberikan himpunan semesta U adalah umur manusia antara [0, 100] . seseorang dikatakan tua jika dia berumur lebih dari 50 tahun sedangkan orang yang berumur 30 tahun atau kurang, dianggap tidak tua. Grafik fungsi keanggotaan untuk

menggambarkan besarnya derajat ketuaan seseorang ditunjukkan pada Gambar II.4, sedangkan fungsi keanggotaanya ditunjukkan oleh rumus 4.1



Gambar II. 2 Fungsi Keanggotaan Tua

$$f(x) = \begin{cases} 0, & x \leq 30 \\ \dfrac{x - 30}{20}, & 30 < x < 50 \\ 1, & x \geq 50 \end{cases}$$

Jika diketahui seseorang memiliki umur 35 tahun, maka dapat diketahui derajat ketuaannya adalah sebesar (35 – 30)/20 yaitu 0,25.

(Kusrini, 2006 ; 27 - 28)

II.2.3 Metode *Tsukamoto*

Pada metode *Tsukamoto*, setiap aturan direpresentasikan menggunakan himpunan-himpunan *fuzzy*, dengan fungsi keanggotaan yang monoton. Untuk menentukan nilai *output crisp*/hasil yang tegas (Z) dicari dengan cara mengubah *input* (berupa himpunan *fuzzy* yang diperoleh dari komposisi aturan-aturan *fuzzy*) menjadi suatu bilangan pada domain himpunan *fuzzy* tersebut. Cara ini disebut

dengan metode defuzzifikasi (penegasan). Metode defuzzifikasi yang digunakan dalam metode *Tsukamoto* adalah metode defuzzifikasi rata-rata terpusat *(Center Average Defuzzyfier).*

II.3. UML

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak.

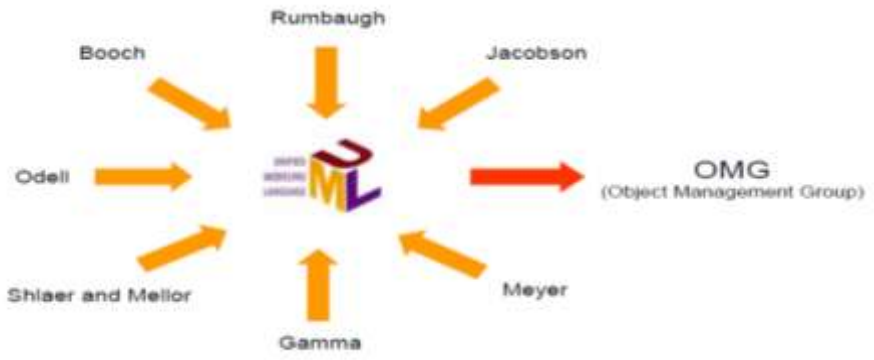
(Sri Dharwiyanti, dan Romi Satria Wahono : 2003 : 2)

Unified Modelling Language (UML) adalah salah satu alat yang sangat handal di dunia pengembangan system yang berorientasi objek.

Munawar (2005 : 17)

UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C. Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan syntax/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML syntax

mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya.



Gambar II.3 *: Metodologi UML*

(Sri Dharwiyanti and Romi Satria Wahono : 2003 : 3)

Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Dimulai pada bulan Oktober 1994 Booch, Rumbaugh dan Jacobson, yang merupakan tiga tokoh yang boleh dikatakan metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease draft pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh Object Management Group (OMG – http://www.omg.org). Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. Booch, Rumbaugh dan Jacobson menyusun tiga buku serial tentang UML pada tahun 1999 [7] [8] [9]. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek.

(Sri Dharwiyanti and Romi Satria Wahono : 2003 : 3)

II.3.1. Konsep Dasar UML

Konsep dasar UML dari berbagai dokumen dan buku UML. Sebenarnya konsepsi dasar UML bisa kita rangkumkan dalam gambar di bawah ini.

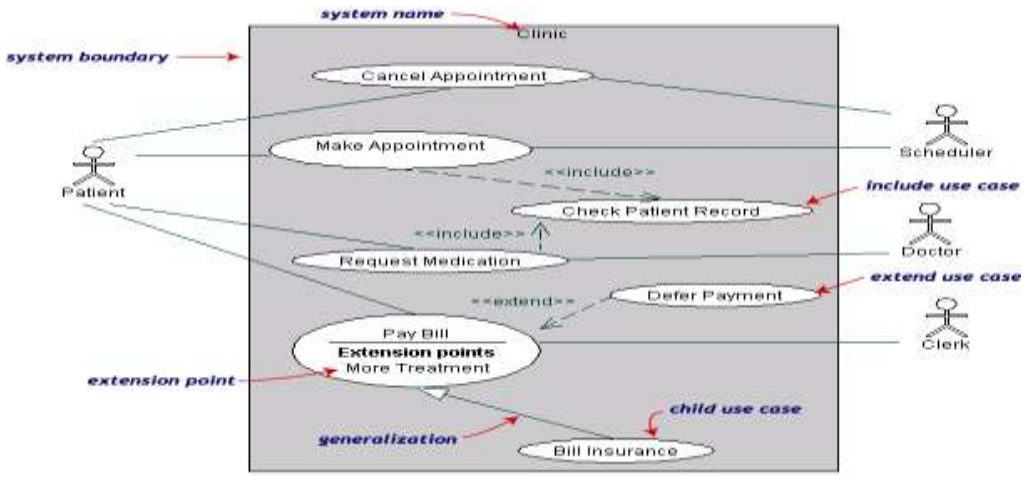
Major Area	View	Diagrams	Main Concepts
structural	*static view*	*class diagram*	*class, association, gene ralization, dependency, realization, interface*
	use case view	*use case diagram*	use case, actor, associat ion, extend, include,use case generalization
	implementation view deployment	*component diagram*	*component, interface, dependency*
	View		*incation*
Dynamic	*state maching view*	*statchart diagram*	*state, event, transition, action*
	activity view	*activity diagram*	*state, activity, completi on transition, lork, join*
	Interaction view	*sequence diagram*	*interaction, object, message, activation*
		collaboration diagram	*collaboration,interactio ,collaboration, role, message*
Model management	*model management view*	*class diagram*	*package, subsystem, model*
extensibility	*All*	*All*	*constraint, stercotype, tagged values*

Tabel II.1: Konsep Dasar UML

(Sri Dharwiyanti and Romi Satria Wahono : 2003 : 3)

II.3.2. *Use case diagram*

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



Gambar II.4. : ***Contoh Diagram Use case***

(Sri Dharwiyanti and Romi Satria Wahono : 2003 : 5)

II. 3.3. Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan

(atribut/properti) suatu bentuk sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut.

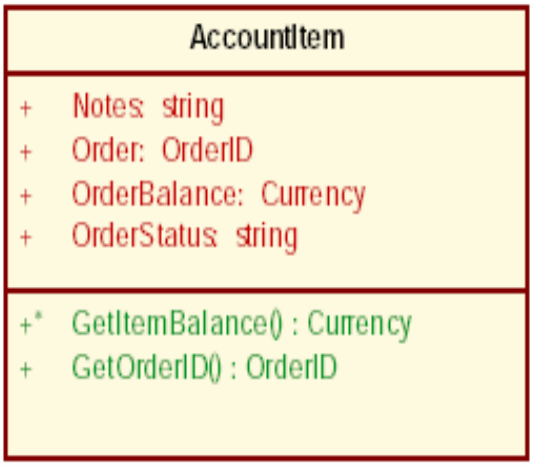
Class diagram menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga area pokok :

1. Nama (*stereotype*)
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

1. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
2. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
3. *Public*, dapat dipanggil oleh siapa saja.

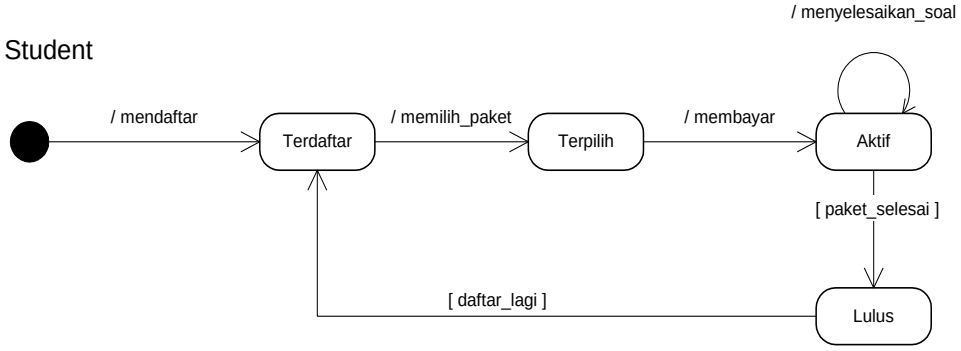


Gambar II.5. *Class Diagram*

(Sri Dharwiyanti and Romi Satria Wahono : 2003 : 5)

II.3.4. *Statechart* Diagram

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu state ke state lainnya) suatu objek pada sistem sebagai akibat dari stimuli yang diterima. Pada umumnya statechart diagram menggambarkan class tertentu (satu class dapat memiliki lebih dari satu statechart diagram).

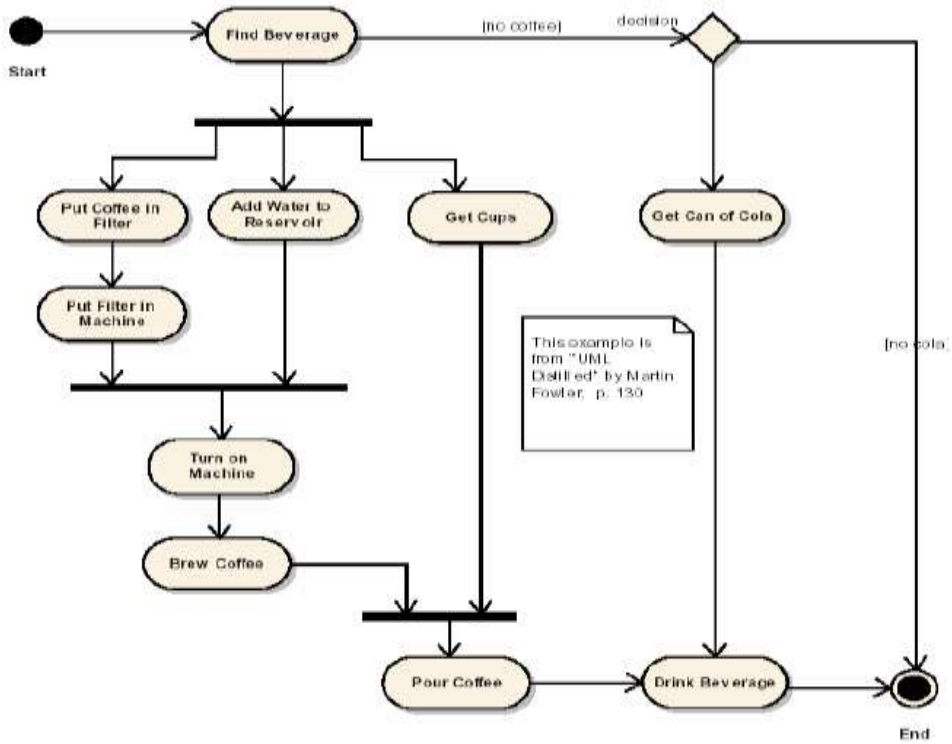


Gambar II.6. ***Contoh Statechart Diagram***

(Sri Dharwiyanti and Romi Satria Wahono : 2003 : 7)

II.3.5. *Activity Diagram*

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



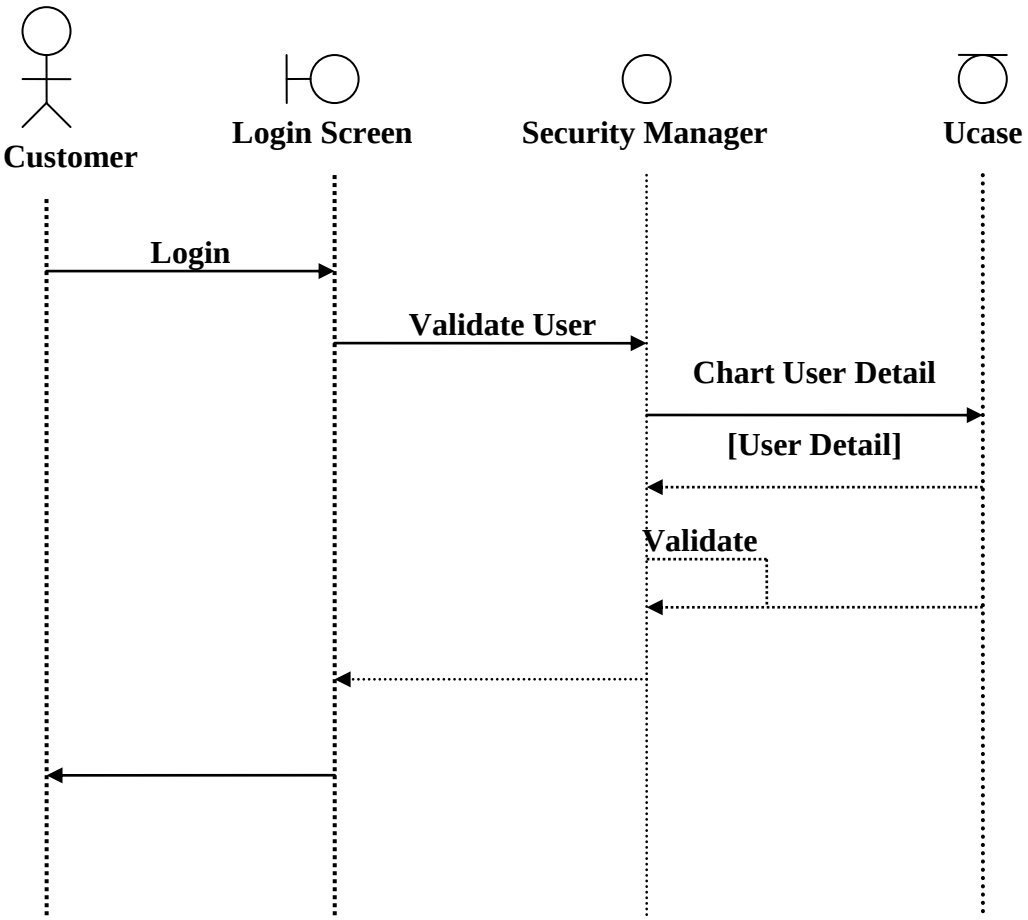
Gambar II.7. Contoh *Activity Diagram* - Tanpa *Swimlane*

(Sri Dharwiyanti and Romi Satria Wahono : 2003 : 8)

II.3.6. *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. Sequence diagram terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan.

Contoh *Sequence Diagram* :



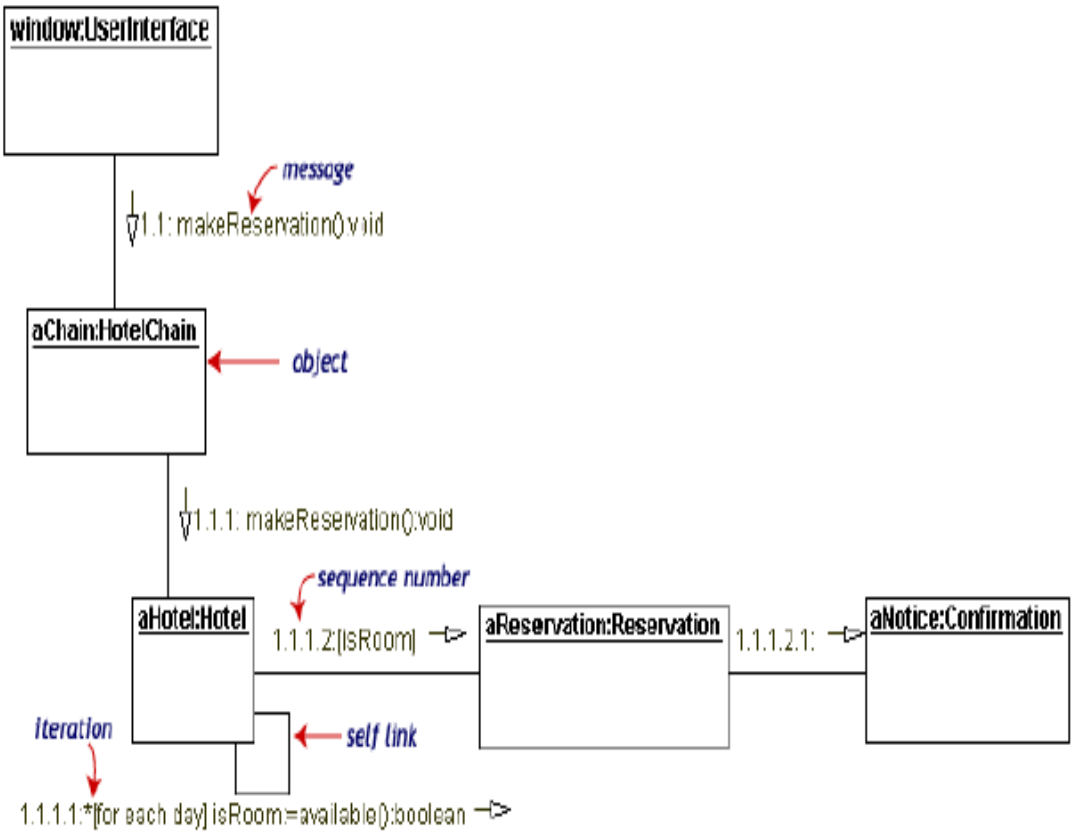
I : Use Case Model

Gambar II.8. *Contoh Sequence Diagram*

(Sri Dharwiyanti and Romi Satria Wahono : 2003 : 9)

II.3.7. *Collaboration Diagram*

Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian message. Setiap *message* memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor 1. *Messages* dari level yang sama memiliki prefiks yang sama.

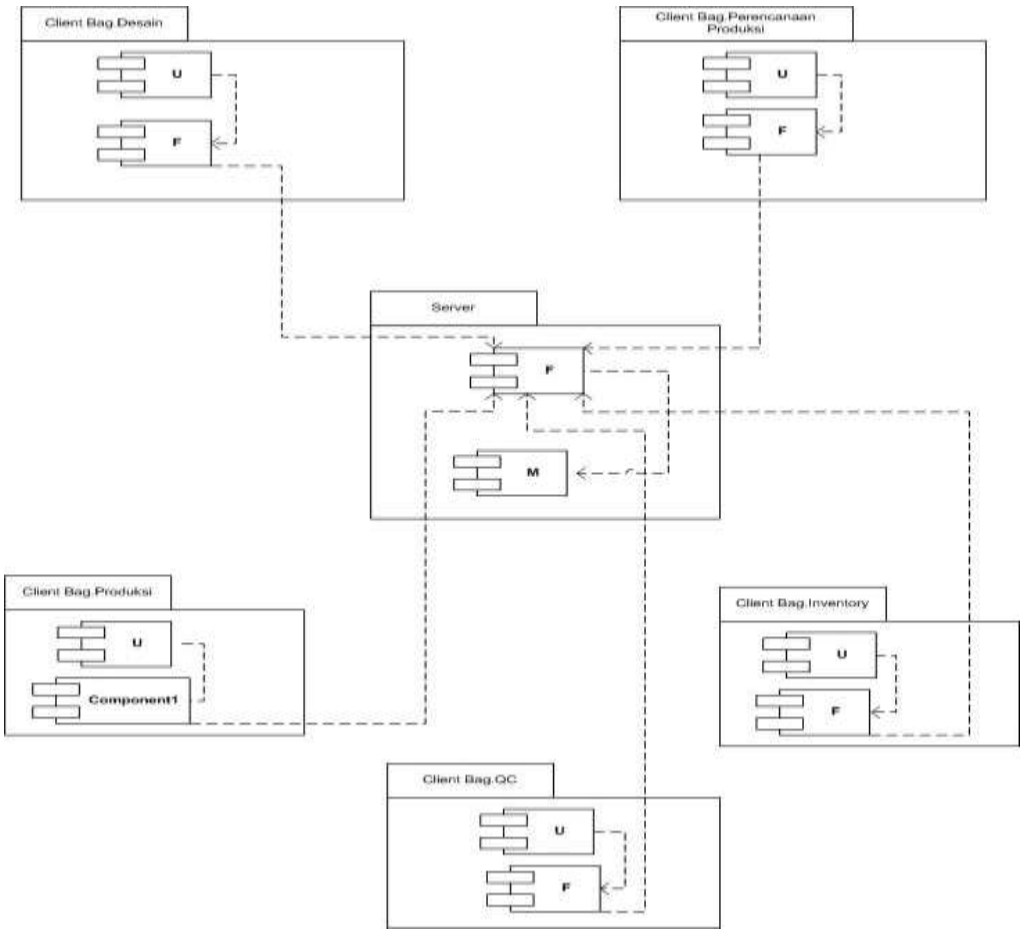


Gambar II.9. ***Contoh Collaboration Diagram***

(Sri Dharwiyanti and Romi Satria Wahono : 2003 : 9)

II.3.8. *Component Diagram*

Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik library maupun *executable*, baik yang muncul pada *compile time*, link time, maupun run time. Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.



Gambar II.9. ***Contoh Component Diagram***

(Sri Dharwiyanti and Romi Satria Wahono : 2003 : 10)

II.4. Database

Database adalah kumpulan data yang saling terkait yang diorganisasi untuk memenuhi kebutuhan dan struktur sebuah organisasi dan dapat digunakan oleh lebih dari satu orang untuk lebih dari satu aplikasi. Ada beberapa konfigurasi yang mungkin dibuat untuk suatu *database.* Pada banyak contoh DSS, data ditempatkan dari data *warehouse* atau sistem database *mainframe legacy* melalui sebuah *server Web* database (lihat DSS dalam Praktik 3.2 dan 3.4). Untuk aplikasi

DSS lainnya, akan disusun *database* khusus jika memang diperlukan. Beberapa database dapat digunakan pada satu aplikasi DSS, tergantung pada sumber data.

(Efraim Turban; 2005 : 145).

II.4.1 *SQL Server 2008*

SQL Server adalah sebuah RDBMS yang kompleks, di mana untuk menyederhanakan aktifitas yang akan dilakukan padanya, maka dibuatlah model. Model itu sendiri berkembang sesuai dengan perkembangan teknologi dengan tujuan menyederhanakan permasalahan dan menambah kemampuan model. Kita telah melihat bagaimana model pengaksesan data berkembang dari ODBC, model DAO, model RDO, dan model ADO.

Aktifitas yang dilakukan oleh Enterprise Manager melalui program yang berisi bermacam – macam fasilitas dinamakan SQL – DMF (*Distributed Management Framework*). Fungsional SQL – DMF dikelompokkan dalam tiga model dasar yaitu:

1. SQL *Namespace* (SQL – NS)
2. SQL *Distributed Management Object* (SQL – MDO)
3. *Distributed Transformation Services* (DTS)

(Harip Santoso : 2006 : 22)

II.4.1.1 DDL

DDL atau *Data Degenition Language* merupakan perintah SQL yang digunakan untuk mendegensisikan atau mendeklarasikan objek database, menciptakan objek database atau bahkan menghapus objek datababse. Objek

database dapat berupa tabel atau database itu sendiri. DDL juga dapat digunakan untuk membuat koneksi antar tabel database beserta batasannya dengan menentukan indeks sebagai kuncinya.

DDL yang umum dipakai adalah :

1. CREATE

Digunakan untuk menciptakan objek database yang baru atau menciptakan database itu sendiri. Untuk membuat database yang baru kita dapat menggunakan perintah berikut:

CREATE DATABASE Nama_Database

Contoh:

CREATE DATABASE Karyawan

2. DROP

Digunakan untuk menghapus objek database. Selain digunakan untuk menghapus database juga digunakan untuk menghapus tabel.

Contoh:

DROP DATABASE Karyawan

DROP DATABASE Mahasiswa

3. ALTER

Digunakan untuk mengubah atribut atau entitas dari objek suatu database. Perintah dasar untuk meng-ubah sebuah tabel menggunakan perintah ALTER adalah sebagai berikut:

ALTER TABLE Nama_Tabel <ACTION>

<ACTION> bisa berupa:

1. Menambah Field

   ADD Nama_Field Tipe_Data

2. Menghapus Field

   DROP Nama_Field

3. Memodifikasi Field

   ALTER Nama_Field TipeData

4. Menambah CONSTRAINT

   ADDCONSTRINT Nama_Const Def_Constraint

5. Menghapus CONSTRAINT

   DROP CONSTRAINT Nama_ Constraint

II.4.1.2 DML

DML atau *Data Manipulation Language* merupakan *query* yang digunakan untuk memanipulasi data, seperti untuk menampilkan data, mengubah data, menghapus data, atau mengisi data. DML yang umum dipakai adalah :

1. SELECT

Merupakan query yang digunakan untuk mengambil data atau menampilkan data. Sintak umum dari SELECT adalah sebagai berikut:

SELECT Daftar_Kolom FROM Nama_Tabel WHERE Kondisi ORDER BY Kolom.

2. INSERT

Perintah DML INSERT digunakan untuk memasukan data ke dalam tabel. Sintaks yang dipakai adalah:

INSERT INTO Nama_Tabel(Daftar_Kolom) VALUE (daftar_Nilai)

Contoh penggunaannya:

INSERT INTO Mahasiswa (Nim,Nama) VALUES (’03.01.1698’,’SUKRISNO’).

(Ema Utami dan Sukrisno : 2005 : 44 -55)

II.4.1.2 DCL

DCL atau *Data Control Language* berisi perintah – perintah untuk mengendalikan pengaksesan data, pengendalian dapat dilakukan berdasarkan per pengguna, per tabel, per kolom maupun per operasi yang boleh dilakukan. Perintah – perintah yang termasuk didalam DCL adalah :

Grant : Memberikan kendali pengaksesan data

Revoke : Mencabut kemampuan pengaksesan data

Lock Table : Mengunci Tabel

(Abdul Khadir : 2006 :107)

II.5. Normalisasi

Normalisasi merupakan cara pendekatan dalam membangun desain logika basis data relasional yang tidak secara langsung berkaitan dengan model data, tetapi dengan menerapka sejumlah aturan dan kriteria standar untuk menghasilkan struktur tabel yang normal. Pada dasarnya desain logika basis data relasional dapat menggunakan prinsip normalisasi maupun transformasi dari model E-R ke bentuk fisik.

Dalam perspektif normalisasi sebuah database dikatakan baik jika setiap tabel yang membentuk basis data sudah dalam keadaan normal. Suatu tabel dikatakan normal, jika:

1. Jika ada dekomposisi/penguraian tabel, maka dekomposisinya di jamin aman (*lossless-join decomposition*)
2. Terpeliharanya ketergantungan funtional pada saat perubahan data (*dependency preservation*)
3. Tidak melanggar *Boyce Code normal Form* (BCNF), jika tidak bisa minimal tidak melanggar bentuk normalisasi ketiga.

II.5.1 Bentuk – Bentuk Normalisasi

1. Bentuk tidak normal

Bentuk ini merupakan kumpulan data yang akan direkam, tidak ada keharusan untuk mengikuti format tertentu, dapat saja tidak lengkap dan terduplikasi. Data dikumpulkan apa adanya sesuai keadaannya.

2. Bentuk normal tahap pertama (1NF)

Defenisi :

Sebuah table disebut 1NF jika :

- tidak ada baris yang duplikat dalam tabel tersebut
- masing – masing *cell* bernilai tunggal

Catatan : Permintaan yang menyatakan tidak ada baris yang duplikat dalam sebuah tabel berarti tabel tersebut memiliki sebuah kunci, meskipun kunci tersebut dibuat dari kombinasi lebih dari satu kolom atau bahkan kunci tersebut merupakan kombinasi dari semua kolom.

Berikut ini akan dicontohkan normalisasi dari tabel kuliah yang memiliki atribut : kode_kul, nama_kul, sks, semester, waktu, tempat, dan nama_dos.

Tabel kuliah tersebut tidak memenuhi normalisasi pertama, karena terdapat atribut waktu yang tergolong ke dalam atribut bernilai banyak. Agar tabel tersebut dapat memenuhi 1NF, maka solusinya adalah dengan mendekomposisi tabel kuliah tersebut :

- Tabel Kuliah (kode_kul, nama_kul, sks, semester, nama_dos)
- tabel Jadwa l(kode_kul, waktu, ruang)

3. Bentuk normal tahap kedua (2NF)

Bentuk Normal Kedua (2NF) terpenuhi jika pada sebuah tabel semua atribut yang tidak termasuk dalam *primary key* memiliki ketergantungan fungsional pada *primary key* secara utuh, sebuah tabel dikatakan tidak memenuhi 2NF jika ketergantunganya hanya bersifat parsial (hanya tergantung pada sebagian dari *primary key*).

Bentuk normal kedua akan dicontohkan berikut :

Misal tabel Nilai terdiri dari atribut kode_kul, nim dan nilai. Jika pada tabel Nilai, misalnya kita tambahkan sebuah atribut yang bersifat redundan, yaitu nama_mhs, maka tabel Nilai ini dianggap melanggar 2NF

Primary keypada tabel Nilai adalah [kode_kul, nim]

Penambahan atribut baru (nama_mhs) akan menyebabkan adanya ketergantungan fungsional yang baru yaitu nim -> nama_mhs. Karena atribut

nama_mhs ini hanya memiliki ketergantungan parsial pada *primary key* secara utuh (hanya tergantung pada nim, padahal nim hanya bagian dari *primary key*). Bentuk normal kedua ini dianggap belum memadai karena meninjau sifat ketergantungan atribut terhadap *primary key* saja.

4. Bentuk normal tahap ketiga (3NF)

Sebuah tabel dikatak memenuhi bentuk normal ketiga (3NF) , jika untuk setiap ketergantungan fungsional dengan notasi X -> A, dimana A mewakili semua atribut tunggal di dalam tabel yang tidak ada di dalam X, maka :

- X haruslah *superkey* pada tabel tersebut.
- atau A merupakan bagian dari *primary key* pada tebel tersebut.

Misalkan pda tabel Mahasiswa, atribut alamat_mhs dipecah ke dalam alamat_jalan, alamat_kota dan kode_pos. Bentuk ini tidak memenuhi 3NF, karena terdapat ketergantungan fungsional baru yang muncul pada tabel tersebut, yaitu :

Alamat_jalan nama_kota -> kode_pos

Dalam hal ini (alamat_jalan, nama_kota) bukan *superkey* sementara kode_pos juga bukan bagian dari primay key pada tabel Mahasiswa, jika tabel Mahasiswa didekomposisi menjadi tabel Mahasiswa dan tabel alamat, maka telah memenuhi 3NF. Hal itu dapat dibuktikan dengan memeriksa dua ketergantungan fungsional pada tabel alamat tersebut, yaitu :

Alamat_jalan nama_kota -> kode_pos

Kode_pos -> nama_kota

Ketergantungan fungsioanal yang pertama tidak melanggar 3NF, karena (alamat_jalan, nama_kota) merupakan *superkey* (sekaligus sebagai *primary key*) dari tabel alamat tersebut. Demikian juga dengan ketergantungan fungsional yang kedua meskipun (kode_pos) bukan merupakan *superkey*, tetapi nama_kota merupakan bagian dari *primary key* dari tabel alamat. Karena telah memenuhi 3NF, maka tebel tersebut tidak perlu di dekompoisi lagi.

5. Bentuk normal tahap keempat (4NF)

Penerapan aturan normalisasi sampai bentuk normal ketiga sudah memadai untuk menghasilkan tabel berkualitas baik. Namun demikian, terdapat pula bentuk normal keempat (4NF) dan kelima (5NF). Bentuk normal keempat berkaitan dengan sifat ketergantungan banyak nilai (*multivalued dependency*) pada suatu tabel yang merupakan pengembangan dari ketergantungan fungsional. Adapun bentuk normal tahap kelima merupakan nama lain dari *Project Join Normal Form* (PJNF).

6. *Boyce Code Normal Form* (BCNF)

    - Memenuhi 1NF
    - Relasi harus bergantung fungsi pada atribut *superkey*.

(Kusrini : 2007 : 40 - 43).

II.6. Kamus Data (*Data Dictionary*)

Kamus data adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi. Dengan kamus data sistem analis dapat mendefinisikan data yang mengalir pada sistem yang lengkap.

Kamus data dibuat dan digunakan baik pada tahap analisis maupun pada tahap perancangan sistem. Pada tahap analisis kamus data digunakan sebagai alat komunikasi antara sistem analis dengan user tentang data yang mengalir pada sistem tersebut serta informasi yang dibutuhkan oleh pemakai sistem (*user*).

Kamus data dibuat berdasarkan arus data yang ada pada data flow diagram. Arus data yang ada di DFD bersifat global dan hanya menunjukkan nama arus datanya saja. Keterangan lebih lanjut tentang struktur dari suatu arus data di DFD dapat dilihat pada kamus data. Kamus data atau data dictionary harus dapat mencerminkan keterangan yang jelas tentang data yang dicatatnya. Untuk keperluan ini maka kamus data harus memuat hal- hal sebagai berikut :

1. Arus Data

Arus data menunjukkan dari mana data mengalir dan kemana data akan menuju. Keterangan arus data ini perlu dicatat di kamus data untuk memudahkan mencari arus data di dalam *data flow diagram* (DFD).

2. Nama Arus Data

Karena kamus data dibuat berdasarkan arus data yang mengalir di DFD, maka nama dari arus data juga harus dicatat dikamus data, sehingga mereka yang

membaca DFD dan memerlukan penjelasan lebih lanjut tentang suatu arus data tertentu di DFD dapat berlangsung mencarinya dengan mudah di kamus data.

3. Tipe Data

Telah dikatahui bahwa arus data dapat mengalir dari hasil suatu proses keproses yang lainnya. Data yang mengalir ini biasanya dalam bentuk laporan serta dokumen hasil cetakan komputer, dengan demikian bentuk dari data yang mengalir dapat berupa dokumen dasar atau formulir. Dokumen hasil cetakan komputer, laporan tercetak, tampilan layar di monitor, variabel, parameter dan field – field. Bentuk data seperti ini perlu dicatat dikamus data.

4. Struktur Data

Struktur data menunjukkan arus data yang dicata pada kamus data yang terdiri dari item – item atau elemen – elemen data.

5. Alias

Alias atau nama lain dari data juga harus dituliskan. Alias perlu ditulis karena data yang sama mempunyai nama yang berbeda untuk orang atau departemen lainnya.

6. Volume

Volume yang perlu dicatat di dalam kamus data adalah volume rata – rata dan volume puncak dari arus data. Volume rata – rata ditunjukkan banyaknya arus data yang mengalir dalam satu periode tertentu sementara volume puncak menunjukkan volume yang terbanyak.

7. Periode

Periode ini menunjukkan kapan terjadinya arus data. Perlu dicatat dikamus data karena dapat digunakan untuk mengidentifikasikan kapan input data harus dimasukkan kedalam sistem. Kapan proses program harus dilakukan dan kapan laporan – laporan harus dihasilkan.

8. Penjelasan

Untuk lebih memperjelas makna dari arus data yang dicatat dikamus data, maka bagian penjelasan dapat diisi dengan keterangan – keterangan tentang arus data tersebut.

(Tata Sutabri : 2005 : 170 - 173)

II.7. *Entity Relationship Diagram*

Perancangan basis data dengan menggunakan model entity relationship adalah dengan menggunakan *Entity Relationship Diagram* (ERD). Terdapat tiga notasi dasar yang bekerja pada model E-R yaitu : *entity sets*, *relationship sets dan attributes*. Sebuah entity adalah sebuah "benda" (*thing*) atau "objek"(*object*) di dunia nyata yang dapat dibedakan dari semua objek lainnya. *Entity sets* adalah sekumpulan entity yang mempunyai tipe yang sama. Kesamaan tipe ini dapat dilihat dari *atribut/property* yang dimiliki oleh setiap *entity*. Misal :

1. Kumpulan orang yang menyimpan uang pada suatu bank dapat didefenisikan sebagai *entity* set nasabah.

2. Kumpulan orang yang belajar di perguruan tinggi didefenisikan sebagai mahasiswa.

   (Kusrini : 2007 : 21)

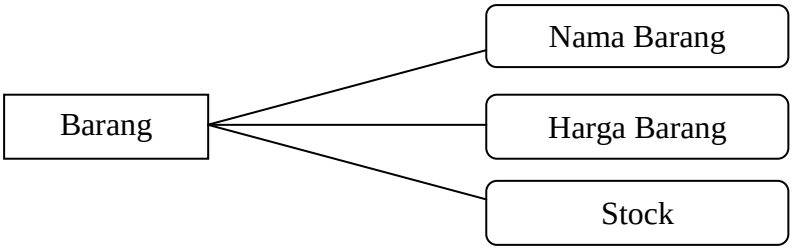
ERD menggunakan sejumlah notasi dan simbol untuk menggambarkan struktur dan hubungan antar data. Pada dasarnya ada 3 macam simbol yang digunakan, yaitu :

1. ***Entity***, adalah suatu objek yang dapat diidentifikasikan dalam lingkungan pemakai, sesuatu yang penting bagi pemakai dalam konteks sistem yang akan dibuat. Sebagai contoh adalah barang, pemasok, pekerja dan lain – lain. Seandainya A adalah barang maka A adalah isi dari barang, sedangkan jika B adalah seorang pelanggan maka B adalah isi dari pelanggan. Karena itu harus dibedakan antara entitas sebagai bentuk umum dari deskripsi tertentu dan isi entitas seperti A dan B dalam contoh diatas, entitas digambarkan dalam bentuk persegi empat.

Gambar II.11 Entitas

(Kusrini dan Andri Koniyo : 2007 : 99)

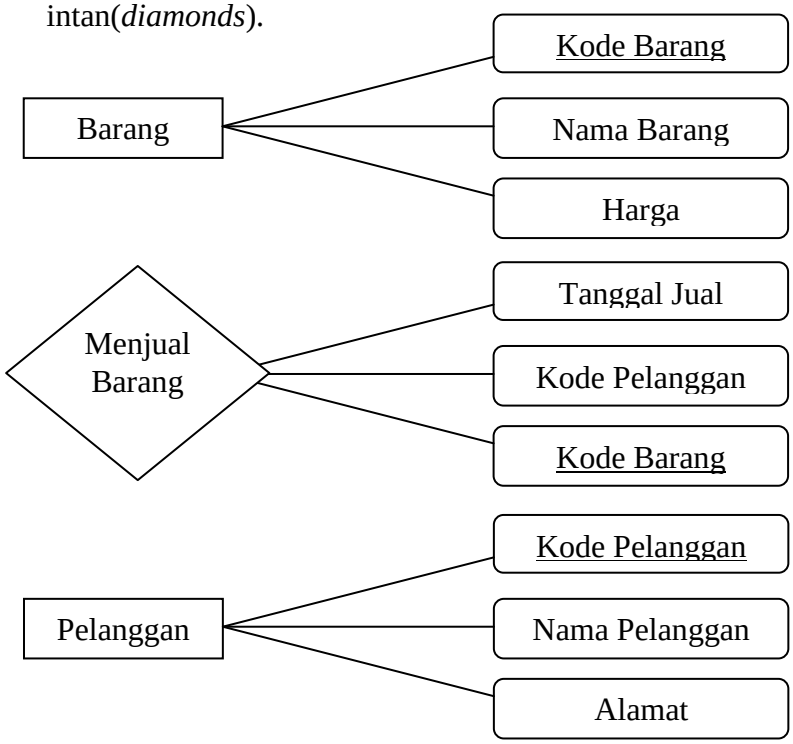
2. ***Atrribut***, entitas mempunyai elemen yang disebut atribut dan berfungsi mendeskripsikan karakter entitas, misalnha atriut nama barang dari entitas barang. Setiap ERD bisa berisi lebih dari satu atribut. Atribut digambarkan dalam bentuk elips.



Gambar II.12 Attribut

(Kusrini dan Andri Koniyo : 2007 : 100)

3. **Hubungan** (***Relationship***), sebagaimana halnya entitas huubungan pun harus dibedakan antara hubungan atau bentuk hubungan antar entitas dengan isi dari hubungan itu sendiri. Misalnya dalam kasus hubungan antara entitas barang dan entitas pelanggan adalah menjual barang, sedangkan isi hubungannya dapat berupa tanggal jual atau yang lainnya. Relationship digambarakan dalam bentuk intan(*diamonds*).



Gambar II.13 *Relationship*

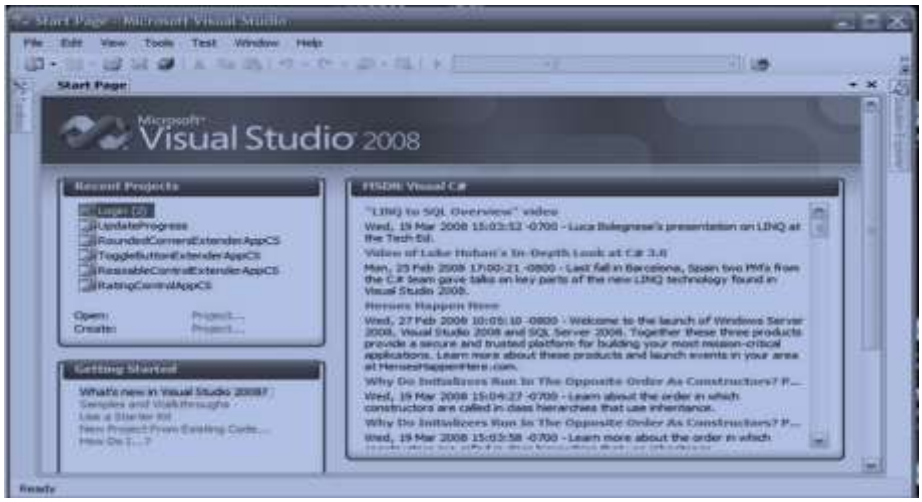
(Kusrini dan Andri Koniyo : 2007 : 100)

II.8. Mengenal Microsoft Visual Studio 2008

Microsoft Visual Studio 2008 merupakan kelanjutan dari *Microsoft Visual Studio* sebelumnya, yaitu *Visual Studio. Net* 2003 yang diproduksi oleh Microsoft. Pada bulan Februari 2002 *Microsoft* memproduksi teknologi. *Net Framework* versi 1.0, teknologi. *Net* ini didasarkan atas susunan berupa *Net Framework*, sehingga setiap produk baru yang terkait dengan teknologi. Net akan selalu berkembang mengikuti perkembangan. *Net Frameworknya*. Pada perkembangannya nantinya mungkin untuk membuat program dengan teknologi. *Net* memungkinkan para pengembang perangkat lunak akan dapat menggunakan lintas sistem operasi, yaitu dapat dikembangkan di sistem operasi windows juga dapat dijalankan pada sistem operasi lain, misalkan pada sistem operasi *Linux*, seperti yang telah dilakukan pada pemograman *Java* oleh *Sun Microsystem*.

Pada saat ini perusahaan-perusahaan sudah banyak meng*update* aplikasi lama yang dibuat *Microsoft Visual Basic* 6.0 ke teknologi. *Net* karena kelebihan-kelebihan yang ditawarkan, terutama memungkinkan pengembang perngkat lunak secara cepat mampu membuat program *robust*, serta berbasiskan integrasi ke internet yang dikenal dengan *XML Web Service* (Ketut Darmayuda ; 2008 : 1)

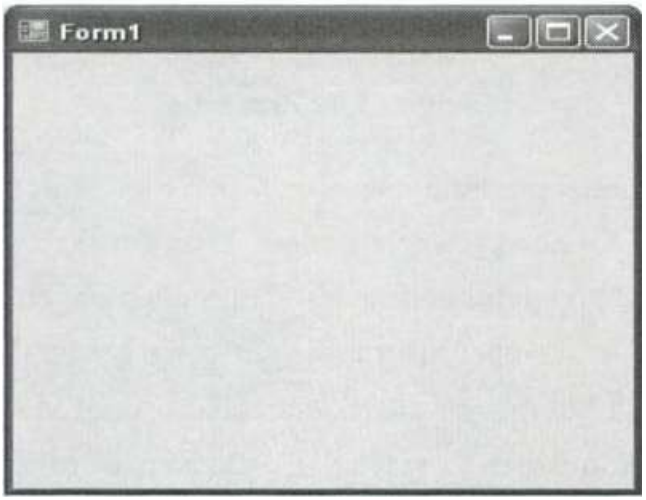
Untuk melihat tampilan visual studio 2008 dapat dilihat pada gambar II.13 sebagai berikut :



Gambar II.14. Tampilan Utama Visual Studio 2008

(Ketut Darmayuda : 2008 : 12)

Tempat membuat aplikasi dan meletakkan komponen yang dibutuhkan dalam aplikasi disebut dengan *form*. *Form* juga digunakan untuk merancang tampilan program aplikasi yang akan di buat. Pada *form* terdapat ikon *Minimize*, *Maximize*, dan *Close*. Apabila mengklik *form*, maka akan tampil titik corner yang apat digunakan untuk memperbesar dan memperkecil lebar *form* sesuai dengan keinginan.

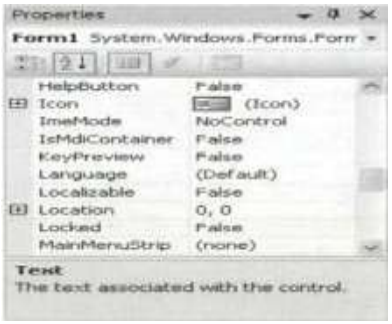


Gambar II.15 *Form*

(Hendra Yudi : 2009 : 9)

Properties digunakan untuk menuliskan atau mengatur *form* dan komponen yang berisi perintah pelengkap dan pengatur aplikasi yang dibuat. Dengan *properties* dapat mengatur warana tulisan, membuat tulisan dengan tebal, miring, atau bentuk lain yang diinginkan. *Properties* juga digunakan sebagai petunjuk perintah yang akan digunakan untuk membuat program dan untuk memanipulasi komponen yang terdapat dalam *form*.

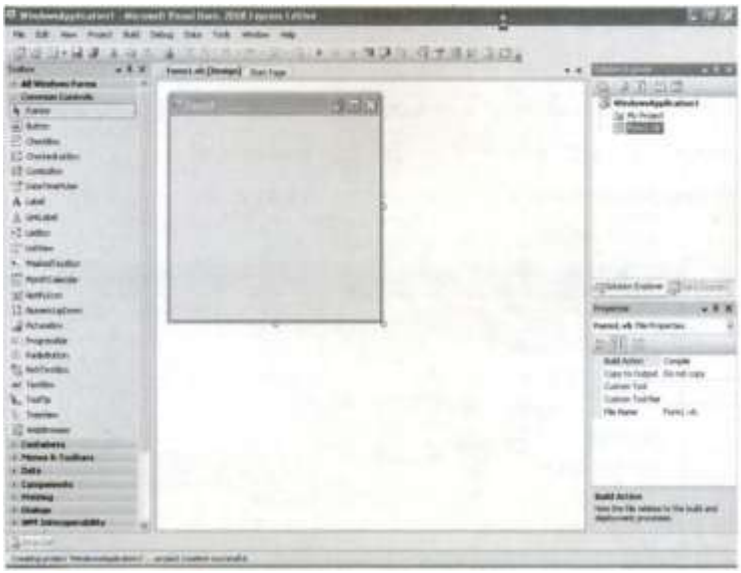
Selain properties juga terdapat *Solution Explorer* yang digunakan untuk menampung informasi *project*, *form*, dan komponen yang aktif pada saat itu. Pada *Solution Explorer* juga terdapat data *source* yang digunakan untuk membuat dan mengolah data dengan berbagai jenis *database*.



Gambar II.16 *Properties*

(Hendra Yudi : 2009 : 10)

Jadi untuk membuat program menggunakan Visual Basic 2008 digunakan *toolbox* yang berisi komponen yang akan digunakan untuk membuat program, kemudian komponen yang sudah dipilih tersebut diletakkan pada *form* untuk membentuk rancangan program aplikasi yang diinginkan. Stelah komponen diletakkan pada *form*, kemudian mengatur properties dari komponen yang telah diletakkan pada *form* aplikasi.



Gambar II.17 Tampilan awal *project* visual basic 2008
(Hendra Yudi : 2009 : 8 – 10)