# BAB II

# TINJAUAN PUSTAKA

## II.1. Sistem

Sistem merupakan sekumpulan elemen-elemen yang saling terintegrasi serta melaksanakan fungsinya masing-masing untuk mencapai tujuan yang telah ditetapkan. Karakteristik sistem terdiri dari :

1. Komponen Sistem

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

2. Batasan Sistem

Batasan merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batasan sistem ini memungkinkan suatu sistem dipandang suatu kesatuan. Batasan suatu sistem menunjukkan ruang lingkup (*scope*) dari sistem tersebut.

3. Lingkungan Luar Sistem

Lingkungan luar dari suatu sistem adalah apapun diluar batas dari sistem yang mempengaruhi operasi sistem. Lingkung luar sistem dapat bersifat menguntungkan dan dapat juga bersifat merugikan sistem tersebut.

4. Penghubung Sistem

Penghubung merupakan media penghubung antara satu subsistem dengan subsistem lainnya. Melalui penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem lainnya.

5. Masukan Sistem

Masukan sistem adalah energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan (*maintanceinput*) dan masukan sinyal (*signalinput*).

6. Keluaran Sistem

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

7. Pengolah 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 (*goal*) atau sasaran (*objective*). Kalau suatu sistem tidak mempunyai sasaran, maka operasi sistem tidak ada gunanya (Jurnal Saintikom ; Sulindawati ; 2010 : 135).

## II.2. Sistem Pakar

Untuk memahami aplikasi sistem pakar, selain memahami definisinya, kita juga harus mengetahui tujuan dari sistem pakar, kompkonen-komponennya,

semua domain, dan contoh-contoh aplikasinya, *stakeholders*, dan alasan digunakannya sistem ini.

Sistem pakar merupakan cabang dari *Artificial interllingence (AI)* yang cukup tua karena sistem ini mulai dikembangkan pada pertengahan 1960. Sistem pakar yang muncul pertama kali adalah *General-purpose problem solver (GPS)* yang dikembangkan oleh Newel dan Simon. Sampai saat ini sudah banyak sistem pakar yang dibuat, seperti MYCIN untuk diagnosa penyakit, DENDRAL untuk mengidentifikasi struktur molekul campuran yang tak dikenal, XCON & XSEL untuk membantu komfigurasi sistem komputer besar, SOPHIE untuk analisis sirkuit elekronik, Prospector digunakan di bidang geologi untuk membantu mencari dan menemukan deposit, FOLIO digunakan untuk membantu memberikan keputusan bagi seorang manager dalam stok dan investasi, DELTA dipakai untuk pemeliharaan lokomotif listrik diesel, dan sebagainya.

Istilah sistem pakar berasal dari istilah *knowledge-based expert system*. Istilah ini mencul karena untuk memecahkan masalah, sistem pakar menggunakan pengetahuan seorang pakar yang dimasukkan ke dalam komputer. Seseorang yang bukan pakar menggunakan sistem pakar untuk meningkatkan kemampuan pemecahan masalah, sedangkan seorang pakar menggunakan sistem pakar untuk *knowledge assistant* (T.Sutojo, dkk ; 2011 : 159-160).

### II.2.1. Manfaat Sistem Pakar

Sistem pakar menjadi sangat populer karena sangan banyak kemampuan dan manfaat yang diberikannya, di anataranya :

1. Meningkatkan produktivitas, karena sistem pakar dapat bekerja lebih cepat daripada manusia.
2. Membuat seorang yang awam bekerja seperti layaknya seorang pakar.
3. Meningkatkan kualitas, dengan memberi nasehat yang konsisten dan mengurangi kesalahan.
4. Mampu menangkap pengetahuan dan kepakaran seseorang.
5. Dapat beroperasi di lingkungan yang berbahaya.
6. Memudahkan akses pengetahuan seorang pakar.
7. Andal, sistem pakar tidak pernah menjadi bosan dan kelelahan atau sakit.
8. Bisa digunakan sebagai meddia pelengkap dalam pelatihan. Pengguna pemula yang bekerja dengan sistem pakar akan menjadi lebih berpengalaman karena adanya fasilitas penjelas yang berfungsi sebagai guru.
9. Meningkatkan kemampuan untuk menyelesaikan masalah karena sistem pakar mengambil sumber pengetahuan dari banyak pakar (T.Sutojo, dkk ; 2011 : 160-161)

### II.2.2. Kekurangan Sistem Pakar

Selain manfaat, ada juga beberapa kekurangan yang ada pada sistem pakar, diantaranya :

1. Biaya yang sangat mahal untuk membuat dan memeliharanya.
2. Sulit dikembangkan karena keterbatasan keahlian dan ketersediaan pakar.
3. Sistem pakar tidak 100% bernilai benar (T.Sutojo, dkk ; 2011 : 161).

### II.2.3. Ciri – Ciri Sistem Pakar

Ciri – ciri dari Sistem Pakar dalah sebagai berikut :

1. Terbatas pada domain keahlian tertentu.
2. Dapat memberikan penalaran untuk data- data yang tidak lengkap atau tidak pasti.
3. Dapat menjelaskan alasan – alas an dengan cara yang dapat dipahami.
4. Bekerja berdasarkan kaidah/ rule tertentu.
5. Mudah dimodifikasi.
6. Basis pengatahuan dan mekanisme inferensi terpisah.
7. Keluarannya bersifat anjuran.
8. Sistem dapat mengaktifkan kaidah secara searah yang sesuai, dituntun oleh dialog dengan pengguna (T.Sutojo,dkk ; 2011: 162).

### II.2.4. Pemindahan Kepakaran

Tujuan dari sistem pakar adalah memindahkan kepakaran dari seorang pakar ke dalam komputer, kemudian ditransferkan kepada orang lain yang bukan pakar. Proses ini melibatkan empat kegiatan, yaitu :

1. Akuisisi pengetahuan (dari pakar atau sumber lain)
2. Representasi pengetahuan (pada komputer)
3. Inferensi pengetahuan
4. Pemindahan pengetahuan ke pengguna (T.Sutojo, dkk ; 2011 : 164).

### II.2.5. Inferensi (*Inferencing*)

Inferensi adalah sebuah prosedur (program) yang mempunyai kemampuan dalam melakukan penalaran. Inferensi ditampilkan pada suatu komponen yang disebut mesin inferensi yang mencakup prosedur-prosedur mengenai pemecahan masalah. Semua pengetahuan yang dimiliki oleh seorang pakar disimpan pada basis pengetahuan oleh sistem pakar. Tugas mesin inferensi adalah mengambil kesimpulan berdasarkan basis pengetahuan yang dimilikinya (T.Sutojo, dkk ; 2011 : 164).

### II.2.6. Aturan – aturan (*Rule*)

Kebanyakan software sistem pakar komersial adalah sistem yang berbasis *rule (rule-based systems),* yaitu pengetahuan disimpan terutama dalam bentuk *rule*, sebagai prosedure-prosedure pemecahan masalah (T.Sutojo, dkk ; 2011 : 165)

### II.2.7. Kemampuan Menjelaskan (*Explanation Capability*)

Fasilitas lain dari sistem pakar adalah kemampuannya untuk menjelaskan saran atau rekomendasi yang berikutnya. Penjelasan dilakukan dalam subsistem yang disebut subsistem penjelasan *(Explanation).* Bagian dari sistem ini memungkinkan sistem untuk memeriksa penalran yang dibuatnya sendiri dan menjelaskan operasi-operasinya (T.Sutojo, dkk ; 2011 : 165).

## II.3. Metode *Certainty Factor* (CF)

*Certainty Theory* ini diusulkan oleh Shortliffe dan Buchanan pada tahun 1975 untuk mengakomadasi ketidakpastian pemikiran (*inexact reasoning*) seorang pakar. Teori ini berkembang bersamaan dengan pembuatan sistem pakar MYCIN. Team pengembang MYCIN mencatat bahwa tim ahli sering kali menganalisa informasi yang ada dengan ungkapan seperti misalnya: mungkin, kemungkinan besar, hampir pasti. Untuk mengakomodasi hal ini tim MYCIN menggunakan *Certainty factor* (CF) guna menggambarkan tingkat keyakinan pakar terhadap masalah yang sedang dihadapi. Secara umum, rule direpresentasikan dalam bentuk sebagai berikut :

IF E1 [AND/OR] E2 [AND/O R] … En

THEN H (CF = CF) (Bain Khusnul ; 2010 : 13).

### II.3.1. Model Perhitungan *Certainty Factor* dengan *Rule*

*Certainty Factor* (CF) menunjukkan ukuran kepastian terhadap suatu fakta atau aturan. Faktor kepastian ini merupakan bentuk penggabungan kepercayaan dan ketidakpercayaan dalam suatu bilangan tunggal. Berikut notasi faktor kepastian:

$CF(Pk, G) = MB(Pk,G) - MD(Pk, G)$

Beberapa *evidence* dapat dikombinasikan untuk menentukan CF dari suatu hipotesis. Untuk sistem ini, tingkat kepastian sistem terhadap kesimpulan yang diperoleh dihitung berdasarkan nilai probabilitas penyakit karena adanya

evident/gejala tertentu [5]. Jika ada gejala dan penyakit sebagai *hipothesis* maka tingkat kepastian diformulasikan sebagai *CF(Pk,G)*:

$$MB(Pk,G)=\begin{cases}1 & ,P(Pk)=1\\ \dfrac{\max[P(Pk|G),P(Pk)]-P(Pk)}{\max[1,0]-P(Pk)} & ,yanglain\end{cases}$$

$$MD(Pk,G)=\begin{cases}1 & ,P(Pk)=1\\ \dfrac{\min[P(Pk|G),P(Pk)]-P(Pk)}{\min[1,0]-P(Pk)} & ,yanglain\end{cases}$$

Di mana:

P(Pk) : probabilitas kerusakan Pk

G : Gejala

CF : Certainty Factor (faktor Kepastian) dalam hipotesis H yang dipengaruhi oleh evidence (fakta) E.

MB : Measure of Belief (tingkat keyakinan), merupakan ukuran kepercayaan dari hipotesis H dipengaruhi oleh evidence (fakta) E.

MD : Measure of Disbelief (tingkat ketidakyakinan) merupakan ukuran ketidakpercayaan hipotesis H dipengaruhi oleh fakta E.

H : Hipotesa atau konklusi yang dihasilkan

Jika ada kaidah lain termasuk dalam hipotesis yang sama tetapi berbeda dalam faktor kepastian, maka perhitungan faktor kepastian dari kaidah yang sama dihitung dari penggabungan fungsi untuk faktor kepastian yang didefinisikan sebagai berikut :

$$CF_{combine}(CF_1, CF_2) = \begin{cases} CF_1 + CF_2(1 - CF_1) & kedua - duanya > 0 \\ \dfrac{CF_1 + CF_2}{1 - \min(|CF_1|, |CF_2|)} & salah\ satu < 0 \\ CF_1 + CF_2(1 - CF_1) & kedua - duanya < 0 \end{cases}$$

Di mana, CFcombine digunakan bergantung pada apakah faktor kepastian positif atau negatif.

Evidensi tidak pasti diperoleh dengan menggali dari hasil wawancara dengan pakar. Nilai CF(Rule)didapat dari interpretasi term dari pakar menjadi nilai MD/MB tertentu. Suatu sistem yang menerapkan *inexact* reasoning, pertama-tama harus menemukan cara untuk mempresentasikan *uncertain evidence*. Pendekatan di atas mengubah bentuk formal dari suatu peluang P(E) dengan CF(E) yang berupa nilai MD/MB.

Faktor kepastian bukanlah suatu peluang, tetapi merupakan ukuran tingkat kepercayaan terhadap suatu evidensi. CF mempresentasikan tingkat kepercayaan bahwa suatu evidensi adalah benar(Bain Khusnul ; 2010 : 13 -14).

## II.4. Kolesistitis

Adanya batu di dalam kandung empedu yang biasanya disertai proses inflamasi. Batu empedu yang terdapat di dalam kandung empedu dapat memberikan gejala nyeri akut episodik akibat kolesistitis akut, kolik bilier, rasa tidak nyaman pada perut yang berulang dan kronik akiba tepisode berulang dari kolik bilier ringan atau gejala-gejala *dyspepsia*.

Tertanamnya batu dalam leher kandung empedu diduga menyebabkan spasme kandung empedu, yang akan menyebabkan kolik bilier. Jika batu jatuh ke belakang, kandung empedu didaerah kosong dan nyeri berhenti, dan jika batu tetap berada di leher kandung empedu akan terjadi nyeri yang terus menerus. Cairan empedu yang terperangkap akan berubah komposisinya menyebabkan inflamasi lokal dan menyebabkan rasa nyeri yang menetap beberapa saat, isi kandung empedu dapat terinfeksi akibat adanya toksemia yang dapat menyebabkan empiema, gangren atau perforasi.

Kontraksi kandung empedu akibat batu adalah penjelasan tradisional terhadap *post prandial discomfort*, tetapi tidak terdapat hubungan yang jelas antara gejala ini dengan adanya batu empedu pada populasi umum. Pada pemeriksaan fisik dapat ditemukan tanda-tanda toksemia, kuadran kanan atas abdomen secara klasik ditemukan*Murphy's sign*. Pada kasus yang lebih lanjut dapat diraba massa inflamasi akibat pembengkakan kandung empedu yang dikelilingi oleh omentum yang melekat. Gambaran klinik berupa demam hilang timbul, takikardia dan gangguan kardiorespirasi merupakan tanda-tanda empiema. Ditemukannya peritonismus difus pada abdomen sebelah atas merupakan tanda perforasi kandung empedu.

Adanya ikterus menunjukkan koledokolitiasis, walaupun kemungkinan *Mirizzi's syndrome,* yaitu akibat kandung empedu yang membengkak, akibat adanya kompresi dari kandung yang disebabkan oleh batu ke duktus koledokus. Kolik bilier dapat memberikan gejala yang sama dengan kolesistitis tetapi

biasanya tidak terpengaruh dengan gerakan dan hanya berlangsung beberapa jam saja. Hal ini sering dipicu oleh makanan berlemak tetapi akan sembuh spontan.

Diagnosis kolelithiasis simptomatik bergantung pada gejala klinisdan terlihatnya batu pada pencitraan. *USG* abdomen untuk meliha tkandung empedu dan saluran empedu adalah tes diagnostik standar untuk pasien kecurigaan batu empedu dan pemeriksaan *USG* ini wajib diperiksa sebelum pasien dioperasi. Jika pasien mengalami serangan kolik bilier berulang dan adanya endapan terdeteksi pada pemeriksaan*USG* maka pasien dianjurkan untuk kolesistektomi (M. Nuhadi ; 2011 : 18-19).

## II.5. SQL Server 2008

SQL Server 2008 adalah sebuah terobosan baru dari Microsoft dalam bidang *database*. SQL Server adalah DBMS (Database Management *System*) yang dibuat oleh Microsoft untuk ikut berkecimpung dalam persaingan dunia pengolahan data menyusul pendahulunya seperti IBM dan Oracle. SQl Server 2008 dibuat pada saat kemajuan dalam bidang *hardware* sedemikian pesat. Oleh karena itu sudah dapat dipastikan bahwa SQL Server 2008 membawa beberapa terobosan dalam bidang pengolahan dan penyimpanan data.

Microsoft merilis SQL Server 2008 dalam beberapa versi yang disesuaikan dengan segment-segment pasar yang dituju. Versi-versi tersebut adalah sebagai berikut. Menurut cara pemrosesan data pada prosesor maka Microsoft mengelompokkan produk ini berdasarkan 2 jenis yaitu :

1. Versi 32-bit(x86), yang biasanya digunakan untuk komputer dengan single prosesor (Pentium 4) atau lebih tepatnya prosesor 32 bit dan sistem operasi Windows XP.
2. Versi 64-bit(x64), yang biasanya digunakan untuk komputer dengan lebih dari satu prosesor (Misalnya Core 2 Duo) dan system operasi 64 bit seperti Windows XP 64, Vista, dan Windows 7.

Sedangkan secara keseluruhan terdapat versi-versi seperti berikut ini:

1. Versi Compact, ini adalah versi "Tipis" dari semua versi yang ada. Versi ini seperti versi desktop pada SQL Server 2000. Versi ini juga digunakan pada handled drvice seperti Pocket PC, PDA, SmartPhone, Tablet PC.
2. Versi Express, ini adalah versi "Ringan" dari semua versi yang ada(tetapi versi ini berbeda dengan versi compact) dan paling cocok untuk latihan para pengembang aplikasi. Versi ini memuat Express Manager standar, integrasi dengan CLR dan XML (Wenny Widya ; 2012 : 3)


**Gambar II.1. Tampilan SQL Server**

***(Sumber : Wenny Widya ; 2012 : 3)***

## II.6. Visual Basic

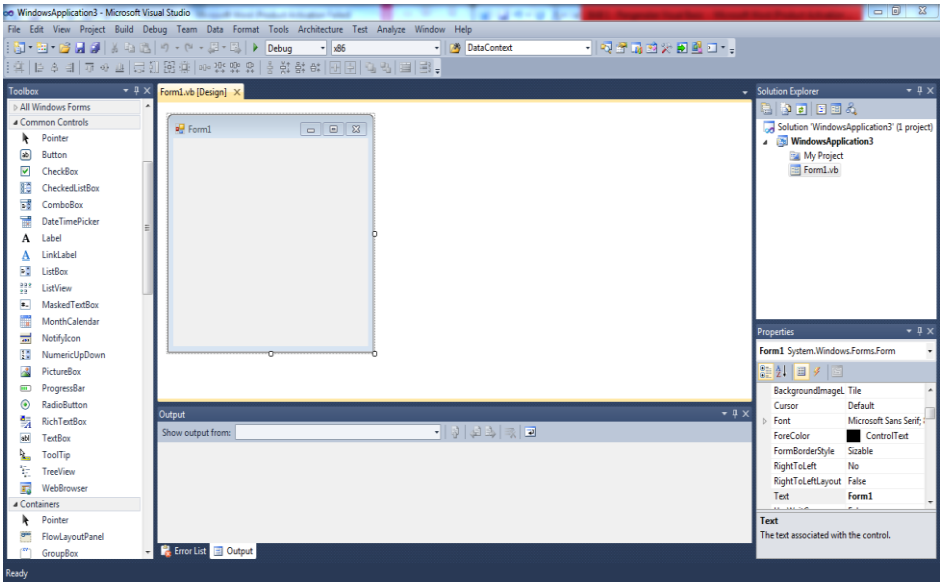
Visual basic dibuat oleh Microsoft, merupakan salah satu bahasa pemrograman beriorientasi objek yang mudah dipelajari. Selain menawarkan kemudahan, Visual Basic juga cukup andal untuk digunakan dalam pembuatan berbagai aplikasi, terutama aplikasi *database.* Visual basic merupakan bahasa pemrograman *event drive,* di mana program aplikasi yang dapat berupa kejadian atau *event*, misalnya ketika *user* mengklik tombol atau menekan enter. Jika kita membuat aplikasi dengan Visual Basic maka kita akan mendapatkan *file* yang menyusun aplikasi tersebut, yaitu :

1. File Project (*.vbp)

   File ini merupakan kumpulan dari aplikasi yang kita buat. File project bisa berupa file *.frm, *.dsr atau file lainnya.

2. File Form (*.frm)

   File ini merupakan file yang berfungsi untuk menyimpan informasi tentang bentuk form maupun *interface* yang kita buat (Edy Winarno ; 2010 : 83).




**Gambar II.2. Tampilan Visual Basic**

***(Sumber : Edy Winarno ; 2010 : 2)***

## II.7. UML (*Unified Modeling Language*)

Menurut Windu Gata (2013 : 4) Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *Unified Modeling Language* (UML). UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak.

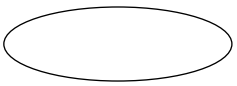
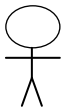
UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem.

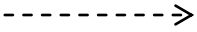
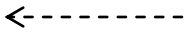
Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

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, yaitu :

**Tabel II.1. Simbol *Use Case***

| Gambar | Keterangan |
|---|---|
|  | *Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan 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*. |

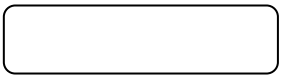
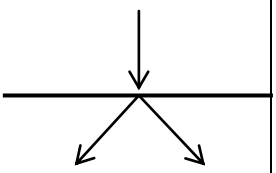
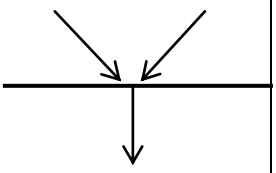
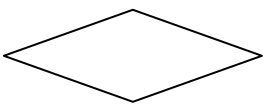
| | |
|---|---|
|  | Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data. |
|  | Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan 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 : Windu Gata ; 2013 : 4*)**

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,* yaitu :

**Tabel II.2. 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.* |

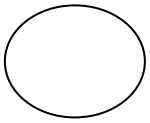
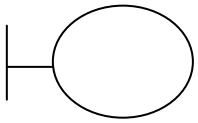
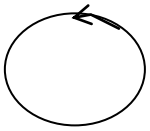
| | |
|---|---|
| New Swimline | *Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa. |

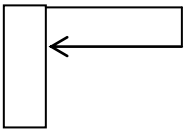
***(Sumber : Windu Gata ; 2013 : 6)***

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*, yaitu :

**Tabel II.3. 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 : Windu Gata ; 2013 : 7)***

4. *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 tanggng 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.

**Tabel II.4.** ***Multiplicity Class Diagram***

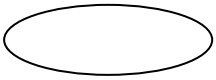
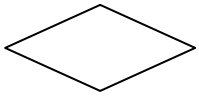
| **Multiplicity** | **Penjelasan** |
|---|---|
| 1 | Satu dan hanya satu |
| 0..* | Boleh tidak ada atau 1 atau lebih |
| 1..* | 1 atau lebih |
| 0..1 | Boleh tidak ada, maksimal 1 |
| n..n | Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4 |

***(Sumber : Windu Gata ; 2013 : 9)***

## II.8. *Entity Relationship Diagram* (ERD)

*Entity Relationship Diagram* atau ERD adalah alat pemodelan data utama dan akan membantu mengorganisasi data dalam suatu proyek ke dalam entitas-entitas dan menentukan hubungan antarentitas. Proses memungkinkan analis menghasilkan struktur basisdata yang baik sehingga data dapat disimpan dan diambil secara efisien (Janner Simarmata ; 2010 : 67).

**Tabel II.5. Simbol ERD**

| Notasi | Keterangan |
|---|---|
|  | persegi panjang mewakili kumpulan entitas |
|  | elips mewakili atribut |
|  | belah ketupat mewakili relasi |
|  | garis menghubungkan atribut dengan kumpulan entitas dan kumpulan entitas dengan relasi |

***(Sumber : Janner Simarmata ; 2010 : 60)***

## II.9. Teknik Normalisasi

Normalisasi adalah teknik perancangan yang banyak digunakan sebagai pemandu dalam merancang basisdata 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 rasional.

Teori normalisasi didasarkan pada konsep bentuk normal. Sebuah tabel relasional dikatakan berada pada bentuk normal tertentu jika tabel memenuhi himpunan batasan tertentu. Ada lima bentuk normal yang tekah ditemukan.

Adapun bentuk – bentuk normalisasi adalah sebagai berikut :

**1. Bentuk normal tahap pertama (1" Normal Form)**

Contoh yang kita gunakan di sini adalah sebuah perusahaan 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.

**2. Bentuk normal tahap kedua ($2^{nd}$ normal form)**

Definisi bentuk normal kedua menyatakan bahwa tabel dengan kunci utama gabungan hanya dapat berada pada 1NF, tetapi tidak pada 2NF. Sebuah tabel relasional berada pada bentuk normal kedua jika dia berada pada bentuk normal kedua jika dia berada pada 1NF dan setiap kolom bukan kunci yang sepenuhnya tergantung pada seluruh kolom yang membentuk kunci utama.

**3. Bentuk normal tahap ketiga ($3^{rd}$ normal form)**

Bentuk normal ketiga mengharuskan semua kolom pada tabel relasional tergantung 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 transitif pada kunci utamanya.

**4. Boyce Code Normal Form (BCNF)**

Setelah 3NF, semua masalah normalisasi hanya melibatkan tabel yang mempunyai tiga kolom atau lebih dan semua kolom adalah kunci. Banyak praktisi berpendapat bahwa menempatkan entitas pada 3NF sudah cukup karena sangat jarang entitas yang berada pada 3NF bukan merupakan 4NF dan 5NF.

**5. Bentuk Normal Tahap Keempat dan Kelima**

Sebuah tabel relasional berada pada bentuk normal keempat (4NF) jika dia dalam BCNF dan semua ketergantungan multivalue merupakan ketergantungan fungsional. Bentuk normal keempat (4NF) didasarkan pada konsep ketergantungan multivalue (MVD).

Sebuah tabel berada pada bentuk normal kelima (5NF) jika ia tidak dapat mempunyai dekomposisi lossless menjadi sejumlah tabel lebih kecil. Empat bentuk normal pertama berdasarkan pada konsep ketergantungan fungsional, sedangkan bentuk normal kelima berdasarkan pada konsep ketergantungan gabungan (*join dependence*) (Janner Simarmata ; 2010 : 76).