

## **BAB II**

### **TINJAUAN PUSTAKA**

#### **II.1. Penelitian Terdahulu**

Berdasarkan penelitian yang dilakukan oleh Istiqomah, dkk (2013) mengenai sistem pakar untuk mendiagnosa penyakit saluran pencernaan menggunakan metode *dempster shafer*, berdasarkan dari hasil penelitian yang sudah dilakukan, kesimpulan yang dapat diambil adalah perangkat lunak yang dihasilkan mampu mendiagnosa penyakit saluran pencernaan pada manusia berdasarkan gejala yang dimasukkan dan dapat memberikan data mengenai penyakit yang diderita berupa nama dan definisi penyakit, penyebab, solusi yang dilengkapi dengan nilai persentase dari penyakit tersebut.

Berdasarkan penelitian yang dilakukan oleh Ashari (2015) mengenai Penerapan Sistem Pakar Untuk Mengidentifikasi Masalah Kehamilan Dengan Metode *Dempster-Shafer*, Kesimpulan yang di dapat adalah perancangan sistem pakar dalam mengidentifikasi masalah kehamilan dilakukan dengan metode *demspter shafer* dengan berdasarkan rule sesuai keahlian para pakar dengan melakukan pengukuran kekuatan *evidence* dalam mendukung suatu himpunan proposisi dan dilakukan oleh sistem komputer sebagai hasil perancangan diagram atau alur sistem.

#### **II.2. Sistem Pakar**

Sistem pakar (*Expert System*) merupakan solusi AI bagi masalah pemrograman pintar (*Intelligent*). Profesor Edward Feigenbaum dari *Stanford*

*University* yang merupakan *pionir* dalam teknologi sistem pakar mendefinisikan sistem pakar sebagai sebuah program komputer pintar (*intelligent computer program*) yang memanfaatkan pengetahuan (*knowledge*) dan prosedur inferensi (*inference procedure*) untuk memecahkan masalah yang cukup sulit hingga membutuhkan keahlian khusus dari manusia. (Rika Rosnelly, 2012).

Sistem pakar (*Expert System*) adalah program berbasis pengetahuan yang menyediakan solusi-solusi dengan kualitas pakar untuk problema-problema dalam suatu *domain* yang spesifik. Sistem pakar merupakan program komputer yang meniru proses pemikiran dan pengetahuan pakar dalam menyelesaikan suatu masalah tertentu. (Dahria, dkk, 2012 : 2).

### **II.2.1. Kelebihan Sistem Pakar**

Sistem pakar memiliki beberapa fitur menarik yang merupakan kelebihannya, seperti :

1. Meningkatnya ketersediaan (*increased availability*). Kepakaran atau keahlian menjadi tersedia dalam sistem komputer. Dapat dikatakan bahwa sistem pakar merupakan produksi kepakaran secara massal (*massproduction*).
2. Mengurangi biaya (*reduced cost*). Biaya yang diperlukan untuk menyediakan keahlian per satu orang *user* menjadi berkurang.
3. Mengurangi bahaya (*reduced danger*). Sistem pakar dapat digunakan di lingkungan yang mungkin berbahaya bagi manusia.

4. Permanen (*permanence*). Sistem pakar dan pengetahuan yang terdapat di dalamnya bersifat lebih permanen dibandingkan manusia yang dapat merasa lelah, bosan, dan pengetahuannya hilang saat sang pakar meninggal dunia.
5. Keahlian *multiple* (*multiple expertise*). Pengetahuan dari beberapa pakar dapat dimuat ke dalam sistem dan bekerja secara simultan dan kontinyu menyelesaikan suatu masalah setiap saat. Tingkat keahlian atau pengetahuan yang digabungkan dari beberapa pakar dapat melebihi pengetahuan yang digabungkan dari beberapa pakar dapat melebihi pengetahuan satu orang pakar.
6. Meningkatkan kehandalan (*increased reliability*). Sistem pakar meningkatkan kepercayaan dengan memberikan hasil yang benar sebagai alternatif pendapat dari seorang pakar atau sebagai penengah jika terjadi konflik antara beberapa pakar. Namun hal tersebut tidak berlaku jika sistem dibuat oleh salah seorang pakar, sehingga akan selalu sama dengan pendapat pakar tersebut kecuali jika sang pakar melakukan kesalahan yang mungkin terjadi pada saat tertekan atau stres.
7. Penjelasan (*explanation*). Sistem pakar dapat menjelaskan detail proses penalaran (*reasoning*) yang dilakukan hingga mencapai suatu kesimpulan. Seorang pakar mungkin saja terlalu lelah, tidak bersedia atau tidak mampu melakukannya setiap waktu. Hal ini akan meningkatkan tingkat kepercayaan bahwa kesimpulan yang dihasilkan adalah benar.

8. Respon yang cepat (*fast response*). Respon yang cepat atau *real time* diperlukan pada beberapa aplikasi. Meskipun bergantung pada *hardware* dan *software* yang digunakan, namun sistem pakar relatif memberikan respon yang lebih cepat dibandingkan seorang pakar.
9. Stabil, tidak emosional, dan memberikan respon yang lengkap setiap saat (*steady, unemotional, and complete response at all times*). Karakteristik ini diperlukan pada situasi *real-time* dan keadaan darurat (*emergency*) ketika seorang pakar mungkin tidak berada pada kondisi puncak disebabkan oleh stress atau kelelahan.
10. Pembimbing pintar (*intelligent tutor*). Sistem pakar dapat berperan sebagai *intelligent tutor* dengan memberikan kesempatan pada *user* untuk menjalankan contoh program dan menjelaskan proses *reasoning* yang dilakukan.

Basis data cerdas (*intelligent database*). Sistem pakar dapat digunakan untuk mengakses basis data secara cerdas. (Rika Rosnelly, 2012).

### **II.2.2. Konsep Umum Sistem Pakar**

Konsep-konsep dasar dari sebuah sistem pakar adalah :

#### **1. Keahlian (*Expertise*)**

Keahlian merupakan pengetahuan khusus yang dimiliki oleh seseorang melalui latihan, belajar, serta pengalaman-pengalaman yang dialami pada suatu bidang tertentu dalam jangka waktu yang cukup lama. Dengan pengetahuan tersebut seorang pakar dapat memberikan keputusan yang lebih baik dan cepat dalam menyelesaikan suatu permasalahan yang sulit.

## 2. Ahli atau pakar (*Expert*)

Seorang pakar harus memiliki kemampuan menyelesaikan permasalahan pada bidang tertentu yang ditanganinya, kemudian memberikan penjelasan mengenai hasil dan kaitannya dengan permasalahan yang ada. Untuk meniru kepakaran seorang manusia, perlu dibangun sebuah sistem komputer yang menunjukkan seluruh karakteristik tersebut. Namun hingga saat ini, pekerjaan dibidang sistem pakar terfokus pada aktifitas penyelesaian masalah dan memberikan penjelasan mengenai solusinya.

## 3. Memindahkan Keahlian (*Transferring Expertise*)

Tujuan dari sistem adalah memindahkan keahlian yang dimiliki oleh seorang pakar ke dalam sebuah sistem komputer.

## 4. Kesimpulan (*Inference*):

Keistimewaan dari sistem pakar adalah kemampuannya dalam memberikan saran, yaitu dengan menempatkan keahlian ke dalam basis pengetahuan (*Knowledge Base*) dan membuat program yang mampu mengakses basis pengetahuan sehingga sistem dapat memberikan kesimpulan. Kesimpulan dibentuk di dalam komponen yang dinamakan mesin pengambil kesimpulan (*Inference Engine*), dimana berisi aturan-aturan untuk menyelesaikan masalah.

## 5. Aturan (*Rule*)

Umumnya sistem pakar adalah sistem berbasis aturan, yaitu pengetahuan yang terdiri dari aturan-aturan sebagai prosedur penyelesaian masalah. Pengetahuan tersebut digambarkan sebagai suatu urutan seri dari kaidah-kaidah yang sudah dibuat.

#### 6. Kemampuan Penjelasan (*Explanation Capability*)

Keistimewaan lain dari sistem pakar adalah kemampuannya dalam memberikan saran atau rekomendasi serta menjelaskan mengapa tindakan tertentu tidak dianjurkan. Pemberian penerangan dan pendapat ini dilakukan dalam suatu subsistem yang dinamakan subsistem penjelasan (*explanation subsystem*). (Sinaga dan Sembiring, 2016 : 96).

### II.2.3. Struktur Sistem Pakar

Komponen yang terdapat dalam struktur sistem pakar ini adalah *knowledge base (rules), inference engine, working memory, explanation facility, knowledge acquisition facility, user interface*. (Rika Rosnelly, 2012)

#### 1. *Knowledge Base* (Basis Pengetahuan)

Basis pengetahuan mengandung pengetahuan untuk pemahaman, formulasi, dan penyelesaian masalah. Komponen sistem pakar disusun atas dua elemen dasar, yaitu fakta dan aturan. Fakta merupakan informasi tentang objek dalam area permasalahan tertentu, sedangkan aturan merupakan informasi tentang cara bagaimana memperoleh fakta baru dari fakta yang telah diketahui.

#### 2. *Inference Engine* (Mesin Inferensi)

Mesin inferensi merupakan otak dari sebuah sistem pakar dan dikenal juga dengan sebutan *control structure* (struktur kontrol) atau *rule interpreter* (dalam sistem pakar berbasis kaidah). Komponen ini mengandung mekanisme pola pikir dan penalaran yang digunakan oleh pakar dalam menyelesaikan suatu masalah. Mesin inferensi disini adalah *processor*

pada sistem pakar yang mencocokkan bagian kondisi dari *rule* yang tersimpan di dalam *knowledge base* dengan fakta yang tersimpan di *working memory*.

### 3. *Working Memory*

Berguna untuk menyimpan fakta yang dihasilkan oleh *inference engine* dengan penambahan parameter berupa derajat kepercayaan atau dapat juga dikatakan sebagai global *database* dari fakta yang digunakan oleh *rule-rule* yang ada.

### 4. *Explanation Facility*

Menyediakan kebenaran dari solusi yang dihasilkan kepada *user* (*reasoning chain*).

### 5. *Knowledge Acquisition Facility*

Meliputi proses pengumpulan, pemindahan dan perubahan dari kemampuan pemecahan masalah seorang pakar atau sumber pengetahuan terdokumentasi ke program *computer*, yang bertujuan untuk memperbaiki atau mengembangkan basis pengetahuan.

### 6. *User Interface*

Mekanisme untuk memberi kesempatan kepada *user* dan sistem pakar untuk berkomunikasi. Antar muka menerima informasi dari pemakai dan mengubahnya ke dalam bentuk yang dapat diterima oleh sistem. Selain itu antarmuka menerima informasi dari sistem dan menyajikannya ke dalam bentuk yang dapat dimengerti oleh pemakai. (Rika Rosnelly, 2012)

### II.2.5. Karakteristik Sistem Pakar

Sistem pakar umumnya dirancang untuk memenuhi beberapa karakteristik umum berikut ini :

1. Kinerja sangat baik (*high performance*). Sistem harus mampu memberikan respon berupa saran (*advice*) dengan tingkat kualitas yang sama dengan seorang pakar atau lebihhinya.
2. Waktu respon yang baik (*adequate respon time*). Sistem juga harus mampu bekerja dalam waktu yang sama baiknya (*reasonable*) atau lebih cepat dibandingkan dengan seorang pakar dalam menghasilkan keputusan. Hal ini sangat penting terutama pada sistem waktu nyata (*real-time*).
3. Dapat diandalkan (*good reliability*). Sistem harus dapat diandalkan dan tidak mudah rusak/ *crash*.
4. Dapat dipahami (*understandable*). Sistem harus mampu menjelaskan langkah-langkah penalaran yang dilakukannya seperti seorang pakar.
5. Fleksibel (*flexibility*). Sistem harus menyediakan mekanisme untuk menambah, mengubah, dan menghapus pengetahuan. (Rika Rosnelly, 2012).

### II.3. Penyakit *Epididimitis*

*Epididimitis* merupakan suatu proses inflamasi yang terjadi pada epididimis. Epididimis merupakan suatu struktur berbentuk kurva (koil) yang menempel di belakang testis dan berfungsi sebagai tempat penyimpanan sperma yang matur. (Buyukcangaz, 2013 : 358). Berikut ini adalah gejala-gejala penyakit *Epididimitis* :

1. Skrotum membengkak, terasa hangat, terasa sakit saat disentuh atau berwarna kemerahan.
2. Nyeri pada testis, biasanya di salah satu sisi. Testis juga bisa terasa sakit saat disentuh.
3. Darah pada cairan sperma.
4. Nyeri saat buang air kecil.
5. Sering ingin buang air kecil dan selalu merasa tidak tuntas.
6. Muncul benjolan disekitar testis.
7. Ujung penis mengeluarkan cairan.

#### **II.4. Metode Dempster Shafer**

Secara umum teori *Dempster-Shafer* ditulis dalam suatu interval:  $[Belief, Plausibility]$ . *Belief* (Bel) adalah ukuran kekuatan *evidence* dalam mendukung suatu himpunan proposisi. Jika bernilai 0 maka mengindikasikan bahwa tidak ada *evidence*, dan jika bernilai 1 menunjukkan adanya kepastian. *Plausibility* (Pls) akan mengurangi tingkat kepastian dari *evidence*. *Plausibility* bernilai 0 sampai 1. Jika yakin akan  $X'$ , maka dapat dikatakan bahwa  $Bel(X') = 1$ , sehingga rumus di atas nilai dari  $Pls(X) = 0$ . (Sinaga dan Sembiring, 2016 : 96).

#### **II.5. Basis Data (Database)**

*Database* adalah sekumpulan tabel-tabel yang saling berelasi, relasi tersebut bisa ditunjukkan dengan kunci dari tiap tabel yang ada. Satu *database*

menunjukkan satu kumpulan data yang dipakai dalam satu lingkup perusahaan atau instansi. *Database* mempunyai kegunaan dalam mengatasi penyusunan dan penyimpanan data, maka seringkali masalah yang dihadapi adalah :

1. Redundansi dan Inkonsistensi data
2. Kesulitan dalam pengaksesan data
3. Isolasi data untuk standarisasi
4. *Multi user*
5. Keamanan data
6. Integritas data
7. Kebebasan data. (Urva dan Siregar; 2015 : 95)

## **II.6. Normalisasi**

Normalisasi merupakan parameter digunakan untuk menghindari duplikasi terhadap tabel dalam basis data dan juga merupakan proses mendekomposisikan sebuah tabel yang masih memiliki beberapa anomali atau ketidakwajaran sehingga menghasilkan tabel yang lebih sederhana dan struktur yang bagus, yaitu sebuah tabel yang tidak memiliki *data redundancy* dan memungkinkan *user* untuk melakukan *insert*, *delete*, dan *update* pada baris (*record*) tanpa menyebabkan inkonsistensi data. (Triyono, 2012 : 19).

### **1. First Normal Form (1 NF)**

Sudah tidak ada *repeating group* yaitu pengulangan yang terjadi pada beberapa atribut atau kolom dalam sebuah tabel, dan juga setiap atribut harus bernilai tunggal. Atribut *multivalued*, *composite*, *derive* tidak tunggal. Setiap nilai dari atribut hanya mempunyai nilai tunggal.

## 2. *Second Normal Form (2 NF)*

Untuk menjadikan tabel normal tingkat ke 2 maka sudah 1NF dan setiap atribut yang bukan *primary key* sepenuhnya secara *functional* tergantung pada semua atribut pembentuk *primary key*.

## 3. *Third Normal Form (3 NF)*

Tabel sudah 2NF dan tidak memiliki *transitive dependencies*, *Transitive dependency* adalah ketika ada atribut yang secara tidak langsung tergantung pada *primary key* dan atribut tersebut juga tergantung pada atribut lain yang bukan *primary key*.

## 4. *Boyce-codd Normal Form (BCNF)*

Tabel dalam BCNF jika sudah 3NF dan semua *determinants* adalah *candidate keys*. Perbedaan 3NF dan BCNF adalah untuk *functional dependency*  $A \rightarrow B$ , 3NF memperbolehkan ketergantungan ada dalam relasi jika B adalah *Primary Key* dan A bukan merupakan *candidate key*. Sedangkan BCNF menuntut untuk ketergantungan tetap ada dalam relasi, A harus menjadi *candidate key*.

## 5. *Fourth Normal Form (4 NF)*

Relasi berada pada bentuk normal keempat apabila memenuhi syarat BCNF dan tidak mempunyai *multivalued dependency*.

## 6. *Fifth Normal Form (5 NF)*

Tabel bentuk normal kelima sering disebut PJNF (*Projection Join Normal Form*), penyebutan PJNF karena untuk suatu relasi akan berbentuk normal

kelima jika tabel tersebut dapat dipecah atau diproyeksikan menjadi beberapa tabel dan dari proyeksi-proyeksi itu dapat disusun kembali (*join*) menjadi tabel yang sama dengan keadaan semula. Jika penyusunan ini tidak mungkin dilakukan dikatakan pada relasi itu terdapat *join dependencies* dan dikatakan bersifat *lossy join*. (Triyono, 2012 : 20).

## **II.7. Microsoft Visual Basic 2010**

*Microsoft Visual Basic* pertama kali diluncurkan pada tahun 1991 dengan nama *Thunder*, yang merupakan *development* pertama yang berbasis *visual* yang dibuat oleh *Microsoft*, untuk menandingi bahasa pemrograman lainnya seperti pemrograman *C*, *C++*, *Pascal*, dan bahasa pemrograman lainnya.

*Visual basic* dibuat oleh *Microsoft*, merupakan salah satu bahasa pemrograman berorientasi 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. (Prayogi, dkk, 2015 : 3).

## **II.8. SQL Server 2008**

*SQL Server 2008* adalah sebuah RDBMS (*Relational Database Management System*) yang sangat powerful dan telah terbukti kekuatannya dalam mengolah data. Dalam versi terbarunya ini, *SQL Server 2008* memiliki

banyak fitur yang bisa diandalkan untuk meningkatkan performa database.

SQL Server 2008 memiliki suatu GUI (*Graphic User Interface*) yang kita gunakan untuk melakukan aktivitas sehari-hari berkaitan dengan database, seperti menulis T-SQL, melakukan *backup* dan *restore database*, melakukan *security database* terhadap aplikasi, dan sebagainya. Pada GUI tersebut kita bisa melakukan settingan terhadap SQL Server untuk bekerja lebih optimal. *Settingan* juga bisa dilakukan menggunakan *script* untuk memudahkan *developer* mengubah *Setting Options* pada SQL Server 2008. (Ruslan, 2013 : 39).

## **II.9. Unified Modeling Language (UML)**

Menurut Windu Gata (2013) 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.


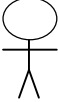
UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem. (Gellysa Urva dan Helmi Fauzi Siregar, 2015, Hal : 93).


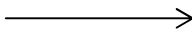
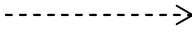
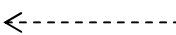
Alat bantu yang digunakan dalam perancangan berorientasi objek berbasis UML adalah sebagai berikut:

### 1. Use case Diagram

*Use case* diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram dapat dilihat pada tabel II.1 dibawah ini:

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

Gambar	Keterangan
	<p><i>Use case</i> menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama <i>use case</i>.</p>
	<p>Aktor adalah <i>abstraction</i> dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasi aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul</p>



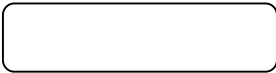
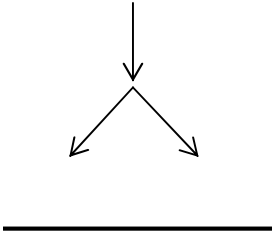
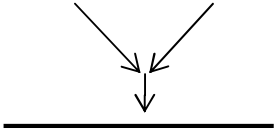
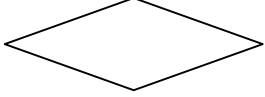

	dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan <i>use case</i> , tetapi tidak memiliki control terhadap <i>use case</i> .
	Asosiasi antara aktor dan <i>use case</i> , digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
	Asosiasi antara aktor dan <i>use case</i> yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	<i>Include</i> , merupakan di dalam <i>use case</i> lain ( <i>required</i> ) atau pemanggilan <i>use case</i> oleh <i>use case</i> lain, contohnya adalah pemanggilan sebuah fungsi program.
	<i>Extend</i> , merupakan perluasan dari <i>use case</i> lain jika kondisi atau syarat terpenuhi.

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

## 2. Diagram Aktivitas (*Activity Diagram*)

*Activity Diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram* dapat dilihat pada tabel II.2 dibawah ini:

Tabel II.2. Simbol *Activity Diagram*

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

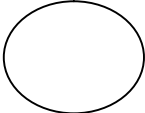
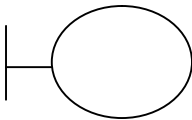
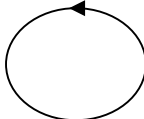
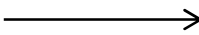
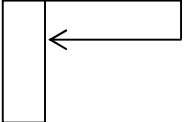

(Sumber : Gellysa Urva dan Helmi Fauzi Siregar; 2015, Hal : 94)


### 3. Diagram Urutan (*Sequence Diagram*)

*Sequence diagram* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima

antar objek. Simbol-simbol yang digunakan dalam *sequence diagram* dapat dilihat pada tabel II.3 dibawah ini :

**Tabel II.3. Simbol *Sequence Diagram***

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

	<p><i>Lifeline</i>, garis titik-titik yang terhubung dengan objek, sepanjang <i>lifeline</i> terdapat <i>activation</i>.</p>
---	--

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

#### 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 tanggung jawab entitas yang menentukan perilaku sistem.

*Class diagram* juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*), *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut.

Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti yang dapat dilihat pada tabel II.4 dibawah ini :

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

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

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