BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terdahulu

Berdasarkan penelitian yang dilakukan oleh Sinaga dan Sembiring (2016) mengenai Penerapan Metode *Dempster Shafer* Untuk Mendiagnosa Penyakit Dari Akibat Bakteri Salmonella**,** Kesimpulan yang di dapat adalah Metode perhitungan kemungkinan *Dempster Shafer* mampu memberikan rekomendasi perhitungan yang akurat untuk dapat dijadikan referensi ketepatan diagnosa untuk mendeteksi penyakit dari akibat Bakteri Salmonella.

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.

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

mendefenisikan 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.

Dengan kata lain, Sistem Pakar adalah sistem komputer yang ditujukan untuk meeniru semua aspek (*emulates*) kemampuan pengambilan keputusan (*decision making*) seorang Pakar. Sistem Pakar memanfaatkan secara maksimal pengetahuan khusus selayaknya seorang pakar untuk memecahkan masalah.

Pakar atau ahli (*expert*) didefenisikan sebagai seseorang yang memiliki pengetahuan atau keahlian khusus yang tidak dimiliki oleh kebanyakan orang. Seorang Pakar dapat memecahkan masalah yang tidak mampu dipecahkan kebanyakan orang. Dengan kata lain, dapat memecahkan suatu masalah dengan lebih efisien namun bukan berarti lebih murah. Pengetahuan yang dimuat ke dalam Sistem Pakar dapat berasal dari seorang pakar atau pun pengetahuan yang berasal dari buku, jurnal, majalah, dan dokumentasi yang dipublikasikan lainnya, serta orang yang memiliki pengetahuan meskipun bukan ahli. Istilah Sistem Pakar (*expert system*). Sering disinonimkan dengan sistem berbasis pengetahuan (*knowledge-based system*) atau Sistem Pakar berbasis pengetahuan (*knowledge based expert system*). (Rika Rosnelly, 2012).

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 masal (*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 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

Pengetahuan yang dimiliki Sistem Pakar direpresentasikan dalam beberapa cara. Salah satu metode yang paling umum digunakan adalah tipe *rule* menggunakan format *IF THEN*. Banyak Sistem Pakar yang dibangun dengan mengekspresikan pengetahuan dalam bentuk *rules*. Bahkan, pendekatan berbasis pengetahuan (*knowledge based approach*) untuk membangun Sistem Pakar telah mematahkan pendekatan awal yang digunakan pada sekitar tahun 1950-an dan 1960-an yang menggunakan teknik penalaran (*reasoning*) yang tidak mengandalkan pengetahuan. (Rika Rosnelly, 2012).

II.2.3. Elemen Manusia Pada Sistem Pakar

Sistem Pakar tidak lepas dari elemen manusia yang terkait di dalamnya. Personil yang terkait dengan sistem pakar ada 4 yaitu :

1. Pakar (*expert*)
2. Pembangun pengetahuan (*knowledge engineer*)
3. Pembangunan sistem (*system engineer*)
4. Pemakai (*user*)

Paling tidak terdapat dua komponen orang atau lebih yang berpartisipasi dalam pembangunan dan penggunaan Sistem Pakar, yakni

sedikitnya seorang pembangun pengetahuan dan seorang pakar. (Rika Rosnelly, 2012).

II.2.3.1. Pakar

Pakar adalah seorang individu yang memiliki pengetahuan khusus, pemahaman, pengalaman, dan metode-metode yang digunakan untuk memecahkan persoalan dalam bidang tertentu. (Rika Rosnelly, 2012)

II.2.3.2. Pembangun/ Pembuat Pengetahuan

Pembuat pengetahuan memiliki tugas utama menerjemahkan dan merepresentasikan pengetahuan yang diperoleh dari pakar, baik berupa pengalaman pakar dalam menyelesaikan masalah maupun sumber terdokumentasi lainnya ke dalam bentuk yang bisa diterima oleh sistem pakar. Dalam hal ini pembangunan pengetahuan (*knowledge engineer*) menginterpretasikan dan merepresentasikan pengetahuan yang diperoleh dalam bentuk jawaban-jawaban atas pertanyaan-pertanyaan yang diajukan pada pakar atau pemahaman, penggambaran analogis, sistematis, konseptual yang diperoleh dari membaca beberapa dokumen cetak seperti *text book*, jurnal, makalah, dan sebagainya. Kurangnya pengalaman *knowledge engineer* merupakan kesulitan utama dalam mengkonstruksi Sistem Pakar. Untuk mengatasi hal tersebut, perancang Sistem Pakar menggunakan *tools* komersial. (Seperti pada editor-editor

khusus maupun *logic debuggers*) dan usahanya akan dipusatkan pada pembangunan mesin inferensi. (Rika Rosnelly, 2012)

II.2.3.3. Pembangun/ Pembuat Sistem

Pembangunan sistem adalah orang yang bertugas untuk merancang antarmuka pemakai Sistem Pakar, merancang pengetahuan yang sudah diterjemahkan oleh pembangun pengetahuan ke dalam bentuk yang sesuai dan dapat diterima oleh Sistem Pakar dan mengimplementasikannya ke dalam mesin inferensi. Selain hal tersebut, pembangun sistem juga bertanggung jawab apabila sistem pakar akan diintegrasikan dengan sistem komputerisasi lain. Alat pembangun (*tool builder*) dapat dipakai untuk menyajikan atau membangun *tool* yang spesifik. Penjual (*vendor*) dapat memberikan *tool* dan saran, staf pendukung dapat memberikan saran dan bantuan secara teknis dalam proses pembangunan Sistem Pakar. (Rika Rosnelly, 2012)

II.2.3.4. Pengguna

Banyak sistem berbasis komputer mempunyai susunan pengguna tunggal. Hal ini berbedaa jauh dengan sistem pakar yang memungkinkan mempunyai beberapa kelas pengguna. Pengguna mungkin tidak terbiasa dengan komputer dan mungkin pada domain masalah. Bagaimanapun juga, banyak solusi permasalahan menjadi lebih baik dan kemungkinan lebih murah dan keputusan yang cepat bila menggunakan Sistem Pakar. Pakar dan pembangun sistem harus

mengantisipasi kebutuhan-kebutuhan pengguna dan membuat batasan-batasan ketika mendesain Sistem Pakar. (Rika Rosnelly, 2012).

II.2.4. 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 oengetahuan 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* (struktu 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 Acquistion 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.

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 melebihinya.
2. 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 GERD

Penyakit Refluks Gastro Esofagus (PRGE) atau yang lebih dikenal dengan nama *Gastro Esophageal Reflux Disease* (GERD) merupakan kondisi yang terjadi bila aliran balik isi lambung ke esofagus memberikan keluhan dan mengganggu kualitas hidup seseorang. GERD dapat memberikan berbagai manifestasi klinis dan komplikasi. Manifestasi dari refluks dapat terjadi di luar

esophagus, salah satunya adalah *laryngopharyngeal reflux (LPR).* Konsep *United Airway Disease* menyatakan bahwa saluran napas atas dan bawah memiliki keterkaitan dalam proses inflamasi sehingga gangguan yang terjadi pada saluran napas atas dapat mempengaruhi fungsi dari saluran napas bawah. (Faisal, 2012 : 1).

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 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 *multivalue 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. *Java*

Java adalah bahasa pemrograman yang multi *platform* dan multi *device*. Sekali anda menuliskan sebuah program menggunakan *Java*. Aplikasi dengan berbasis *Java* ini dikompulasikan ke dalam *p-code* dan bisa dijalankan dengan *Java Virtual Machine*. Fungsionalitas dari *Java* dapat berjalan dengan *platform* sistem operasi yang berbeda karena sifatnya yang umum. (Hardianto dan Handaga, 2015 : 4).

II.8. MySQL

MySQL adalah *Relation Database Management System* (RDBMS) yang didistribusikan secara gratis di bawah lisensi GPL (*General Public License).* MySQL merupakan turunan dari salah satu konsep utama dalam *database* sejak lama, yaitu SQL (*Structure Query Language)*. SQL merupakan salah satu konsep pengoperasian *database*, terutama sebagai seleksi dan pemasukan data, yang memungkinkan pengoperasian datanya dikerjakan dengan mudah secara otomatis. (Inayah, dkk, 2015 : 5).

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.

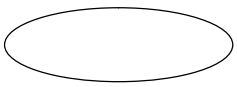
UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem. (Gellysa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

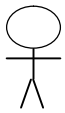
Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut:

1. *Use case* Diagram

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

Tabel II.1. Simbol *Use Case*

Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.

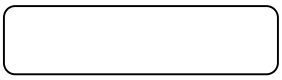
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran.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

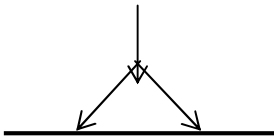
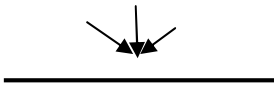
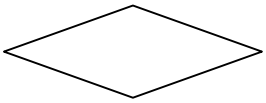
(Sumber : Gellysa Urva dan Helmi Fauzi Siregar; 2015, Hal : 94)

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
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.

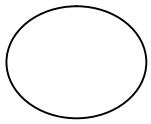
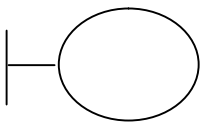
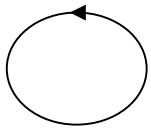
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true, false.*
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

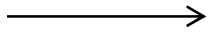
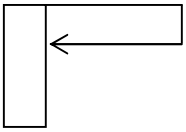
(Sumber : Gellysa Urva dan Helmi Fauzi Siregar; 2015, Hal : 94)

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
	*Entity Class*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*Boundary Class*, berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form* cetak.
	*Control class*, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.

	*Message*, simbol mengirim pesan antar *class*.
	*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation*, *activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

(Sumber : Gellysa Urva dan Helmi Fauzi Siregar; 2015, 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 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 yang dapat dilihat pada tabel II.4 dibawah ini :

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