BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terkait

Adapun penelitian terkait yang akan digunakan sebagai sumber acuan yang relevan dan terkini yaitu:

Lido sabda lesmana (2015) dengan judul Sistem Pakar Dengan Metode Forward Chaining Untuk Diagnosa Pasien Yang Terinfeksi Virus *MERS* Cov (Studi Kasus Di Rsup M.Djamil Padang) dimana penelitian tersebut menghasilkan sebuah sistem pakar dengan hanya menggunakan metode forward chaining dimana hasil diagnosa hanya berdasarkan jawaban ya atau tidak dan tanpa ada perhitungan lainnya.

Aulia Rahman Fahindra (2020) dengan judul Sistem Pakar Dengan Metode Forward Chaining Untuk Diagnosa Pasien Yang Terinfeksi Virus *MERS* Cov (Studi Kasus Di Rsup M.Djamil Padang) dimana penelitian tersebut menghasilkan sebuah sistem yang dapat mendiagnosa hanya penyakit Covid 19 dengan metode *Certainty Factor*..

Sari Murni (2018) dengan judul "Penerapan Metode Teorema Bayes Pada Sistem Pakar Untuk Mendiagnosa Penyakit Lambung" dan hasil penelitian dari penelitian yaitu : Peneliti mencoba memberikan salah satu solusi yang dapat dilakukan untuk membantu para penderita pnyakit lambung dalam mengidentifikasi penyakit tersebut. Pada penelitian ini peneliti menerapkan *teorema Bayes* untuk menghitung nilai probabilitas hasil identifikasi penyakit

lambung. Pada pengujian sampel data gejala penyakit menunjukkan bahwa menghasilkan nilai akurasi sebesar 90 %.

Rame R Girsang (2019) dengan judul "Sistem Pakar Diagnosa Penyakit Mata Katarak dengan Metode *Certainty Factor* Berbasis Web " dan hasil penelitian dari penelitian ini yaitu : dengan menggunakan data mampu berjalan baik secara fungsional untuk mendiagnosa penyakit mata katarak menggunakan metode *Certainty Factor,* yang dapat memberikan kepastian kepada user akan penyakit dan solusi yang diberikan oleh sistem.

Sari Noorlima Yanti (2020) dengan judul "Aplikasi Sistem Pakar untuk Mendiagnosa Virus *Covid-19* pada Manusia Berbasis Web Menggunakan Metode Forward Chaining" dan hasil penelitian dari penelitian ini yaitu : Berdasarkan hasil penelitian dan pembahasan dapat ditarik kesimpulan bahwa keakuratan sistem yang dilakukan terhadap pengujian sistem yang dilakukan oleh 23 pasien terdapat 22 kasus yang sesuai dan 1 kasus yang tidak sesuai. Jadi hasil pengujian sistem dari 23 pasien menghasilkan tingkat akurasi sebesar 96%.

Dari kelima jurnal tersebut peneliti akan menciptakan sebuah sistem pakar yang berbeda dimana dapat melakukan diagnosa dari ketiga penyakit pernafasan *SARS*, *MERS* dan *Covid-19* dengan perhitungan nilai persentase menggunakan metode *Certainty Factor* dan *Teorema Bayes* dan membandingkan kedua hasil perhitungannya.

II.2. Landasan Teori

II.2.1. Sistem Pakar

Sistem pakar adalah cabang kecerdasan buatan yang menggunakan pengetahuan/*knowledge* khusus untuk memecahkan masalah pada level human *expert*/pakar Sistem pakar banyak dikembangkan dalam berbagai ilmu, salah satu diantaranya dalam bidang kedokteran untuk melakukan diagnosa penyakit. Sistem pakar digunakan untuk menentukan diagnosa penyakit akan membantu mengkonfirmasi diagnosa dan menentukan saran dan terapinya.

Sistem pakar adalah sistem in*form*asi berbasis komputer yang menggunakan pengetahuan pakar untuk mencapai per*form*a keputusan tingkat tinggi dalam domain persoalan sempit. Bagian dalam sistem pakar terdiri dari 2 komponen utama yakni berisi *knowledge base* yang berisi basis pengetahuan dan mesin inferensi yang mengGambarkan kesimpulan. Kesimpulan tersebut merupakan respon dari sistem pakar atas permintaan pengguna. (Elida Tuti Siregar : 2015, 2)

II.2.1.1. Konsep Sistem Pakar

Konsep-konsep sistem pakar adalah:

1. Keahlian *(Expertise)*

Keahlian yaitu pengetahuan khusus yang dimiliki oleh seseorang berdasarakan pengalaman-pengalaman yang dialami selama ini pada suatu bidang tertentu dalam jangka waktu yang panjang dengan cara belajar dan latihan. Dengan pengetahuan tersebut, seorang pakar dapat membuat suatu keputusan

dengan mudah, cepat, dan lebih baik 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 menunjukan seluruh karakteristik tersebut. Namun hingga saat ini, pekerjaan dibidang sistem pakar terfokus pada aktifitas penyelesaian masalah dan memberikan penjelasan mengenai solusinya.

3. Memindahkan Keahlian (*Transfering Expertise*)

sistem bertujuan untuk memindahkan keahlian yang dimiliki oleh seorang pakar ke dalam sebuah sistem komputer, kemudian dari sebuah sistem komputer kepada orang lain yang bukan pakar. Ada empat kegiatan pada proses ini :

a. Perolehan pengetahuan (*Knowledge Acquistion*).

b. Representasi pengetahuan (*Knowledge Representation*).

c. Menyimpulkan pengetahuan (*Knowledge Inferencing*).

d. Memindahkan pengetahuan kepada pemakain (*knowledge Transfer to User*).

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 sehinggga 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*)

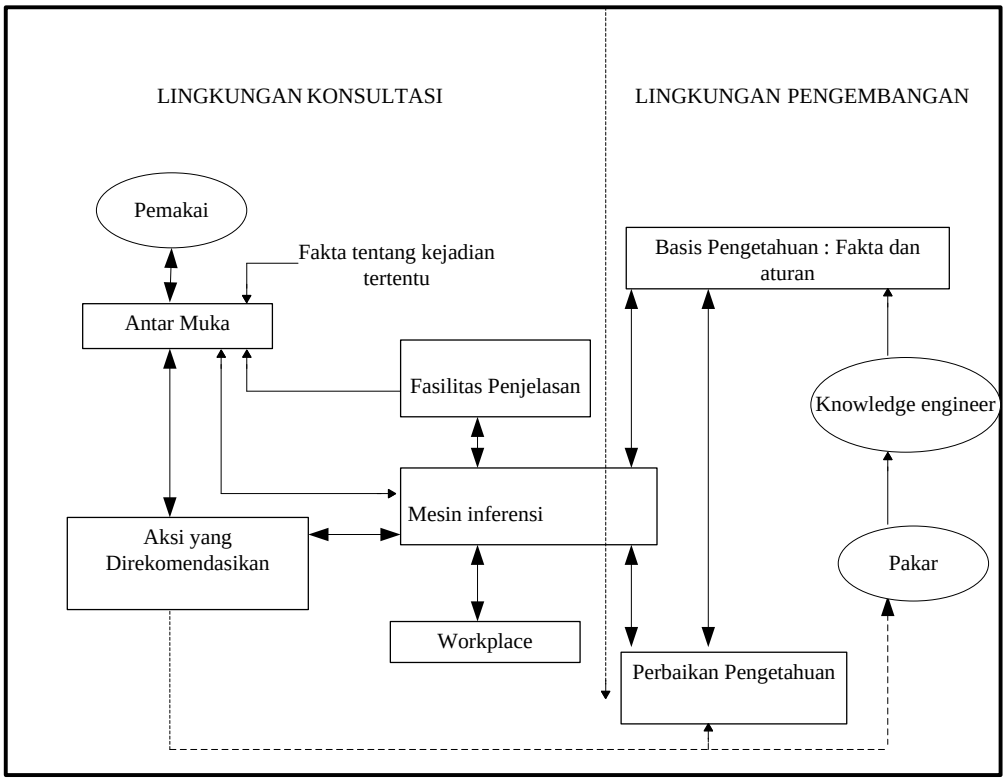
Umumnya sistem pakar adalah sistem berbasis aturan, yaitu pengetahuan yang terdiri dari aturan-aturan sebagai prosedur penyelesaian masalah. Pengetahuan tersebut diGambarkan sebagai suatu urutan seri dari kaidah- kaidah yang sudah dibuat.

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*).(Fithry Tahel, 2018).

II.2.1.2. Arsitektur Sistem Pakar

Sistem pakar disusun oleh dua bagian utama, yaitu Lingkungan Pengembangan dan Lingkungan Konsultasi. Lingkungan Pengembangan digunakan untuk memasukkan pengembangan pakar ke dalam lingkungan sistem pakar. Lingkungan konsultasi digunakan oleh nonpakar untuk memperoleh pengetahuan dan nasehat pakar. Kebanyakan sistem pakar saat ini tidak berisi komponen perbaikan pengetahuan (Innova Siahaan : 2018).



Gambar II.1. Arsitektur Sistem pakar
(sumber : Made Agung Raharja, 2016)

II.2.2. Metode *Certainty Factor*

Faktor kepastian (*Certainty Factor*) 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. Tim pengembang MYCIN mencatat bahwa dokter sering kali menganalisa in*form*asi yang ada dengan ungkapan seperti misalnya: mungkin, kemungkinan besar, hampir pasti, dan sebagainya. Untuk mengakomodasi hal ini tim MYCIN. menggunakan *certainty factor* (CF) guna mengGambarkan tingkat keyakinan pakar terhadap masalah yang sedang dihadapi. Rumus umum menentukan *certainty factor* :

CF[H,E] = MB[H,E] – MD[H,E]

dengan :

CF[h,e] = faktor kepastian

MB[h,e] = ukuran kepercayaan terhadap hipotesis h, jika diberikan *evidence* e (antara 0 dan 1)

MD[h,e] = ukuran ketidakpercayaan terhadap *evidence* h, jika diberikan *evidence* e (antara 0 da1)

II.2.3. Metode *Teorema Bayes*

Teorema bayes merupakan satu metode yang digunakan untuk menghitung ketidakpastian data menjadi data yang pasti dengan membandingkan antara data ya dan tidak. Probabilitas *bayes* merupakan salah satu cara untuk mengatasi ketidakpastian data dengan menggunakan *form*ula *bayes* yang dinyatakan : (Elida Tuti Siregar : 2015)

Teori *bayes* didasarkan pada prinsip bahwa jika terjadi tambahan in*form*asi atau *evidence*, maka nilai probabilitas dapat diperbaiki. Oleh karena itu, teori ini bermanfaat untuk mengubah atau memperbaiki nilai kemungkinan yang ada menjadi lebih baik dengan didukung in*form*asi atau evidence tambahan. Dalam bidang kedokteran *teorema bayes* sudah dikenal tetapi teorema ini lebih banyak diterapkan dalam logika kedokteran modern. *Teorema* ini lebih banyak diterapkan pada hal-hal yang berkenaan dengan diagnosis secara *statistic* yang berhubungan dengan probabilitas serta kemungkinan dari penyakit dan gejala-gejala yang berkaitan.

$$P(H|E) = \frac{P(E|H).P(H)}{P(E)}$$

Dimana :

$P(H|E)$ = probabilitas hipotesis H jika diberikan *evidence* E

$P(E|H)$ = probabilitas munculnya *evidence* E jika diketahui hipotesis H

P(H) = probabilitas H tanpa mengandung *evidence* apapun

P(E) = probabilitas *evidence* E (Elida Tuti Siregar : 2015)

II.2.4. *Visual Studio* 2010

Visual studio adalah *software* untuk mengembangkan apikasi. Itu artinya, kalau ingin membuat program, *software*, aplikasi, dan bahkan *mobile app*, maka bisa menggunakan *visual studio*. Dengan menggunakan *visual studio* bisa menulis kode-kode program, menjalankan kode program, melakukan pengujian, *debugging*, mengemas menjadi aplikasi mandiri, dan banyak lagi. Jadi, dapat diibaratkan dengan sebutan yang lebih mudah, *visual studio* adalah ‘pabrik’ untuk pembuatan *software.*

Dari sini muncul istilah yang disebut IDE yang merupakan singkatan dari *Integrated Development Environment*. Jika diterjemahkan secara bebas, maka *visual studi*o adalah *software* yang menyediakan lingkungan bagi pengembangan aplikasi yang teritegrasi dari hulu ke hilir. *Visual studio* sendiri dibuat oleh *Microsoft* sehingga berorientasi pada MS *Windows*. (Gregorius Agung, 2019 : 3)

Pemrograman *Visual Basic .NET* adalah bahasa pemrograman terpopuler. Ini merupakan pemrograman yang berjalan di atas *platform NET Framework*. Karena itu setiap kali pemrograman *VB.NET* ini merilis versi barunya, tentu saja akan diikuti atau berbarengan dengan perkembangan *Net Framework* terbaru. Framework adalah software yang berisi library yang amat banyak serta menyediakan interoperabilitas bahasa pemrograman. (Edy Winarno, 2015 : 17-18)

Visual Studio 2010 memiliki lebih dari satu computer, SDK (*Software Development Kit*), dan Dokumentasi Tutorial (MSDN *Library).* Komputer yang dimasukkan kedalam *Visual Studio* 2010 antara lain *Visual Basic, Visual* C# , *Visual* C++, *Visual InterDev, Visual* J+, *Visual* F#, dan *Visual Source Safe*, dan banyak lagi laiannya. Semua itu sudah terpaket dan diperuntukkan kedalam *platform .Net Framework* 4.0 atau versi yang lebih tinggi. *Visual studio* ini digunakan untuk membuat aplikasi yang berbasis *dekstop* yang merupakan *platform windows,* namun juga dapat dijalankan dalam bentuk *Microsoft Intermediate Language* diatas *.Net Framework*. Selain itu *Visual studio* juga dapat digunakan untuk membuat aplikasi yang dapat dijalankan diatas *windows mobile* yang berjalan diatas *.Net Compact Framework.*

Visual Studio 2010 terbagi menjadi beberapa tipe diantaranya :

1. *Visual Studio* 2010 *Express Edition* yang biasa digunakan secara gratis tanpa memeberikan *royalty* kepada *Microsoft Inc*.
2. *Visual Studio Standard Edition.*
3. *Visual Studio* 2010 *Professional Edit.*
4. *Visual Studio* 2010 *Ultimate Edition*. (Yesputra Rolly, 2017)

II.2.5. *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 yag 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 :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*).

a. Versi 64-bit(x64), yang biasanya digunakan untuk komputer dengan lebih dari satu prosesor (Misalnya *Core* 2 Duo) dan sistem operasi 64 bit seperti *Windows XP* 64, *Vista, dan Windows* 7. Sedangkan secara keseluruhan terdapat versi-versi seperti berikut ini :
b. Versi *compact*, ini adalah versi "Tipis" dari semua yang ada. Ini seperti versi *desktop* pada *SQL Server* 2000. Versi ini juga digunakan pada *handled device* seperti *pocket PC*, PDA, *Smartphone, Tablet PC.*
c. Versi *Express*, ini adalah versi "Ringan" dari semua yang ada (tetapi versi ini berbeda dengan versi *compact*) dan paling cocok untuk latihan. *Express*

Manager standar, integrasi dengan CLR dan XML. (Agus Tinus Satiawan, 2016)

II.2.6. *UML (Unified Modelling Language)*

Unified Modelling Language (UML) adalah sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak sebuah sistem. UML lebih mengedepankan penggunaan *Diagram* untuk mengGambarkan aspek dari sistem, karena tergolong bahasa visual yang lebih mudah dan lebih cepat dipahami dibandingkan dengan bahasa pemrograman. *Unified Modelling Language* (UML) biasa digunakan untuk :

1. MengGambarkan batasan sistem dan fungsi-fungsi sistem secara umum, dibuat dengan *use case* dan *actor*.
2. MengGambarkan kegiatan atau proses bisnis yang dilaksanakan secara umum, dibuat dengan *interaction Diagrams*.
3. MengGambarkan representasi struktur static sebuah sistem dalam bentuk *class Diagram*.
4. Membuat model *behavior* yang mengGambarkan kebiasaan atau sifat sebuah sistem dengan *state transition Diagrams* UML.
5. Menyatakan arsitektur implementasi fisik menggunakan *component and development Diagrams.*
6. Menyampaikan atau memperluas *fungsionalty* dengan *stereo types*.

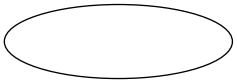
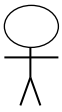
Pemodelan penggunaan UML merupakan metode pemodelan berorientasi objek dan berbasis visual. Karenanya pemodelan objek yang fokus pada pendefinisian struktur statis dan model sistem in*form*asi yang dinamis daripada mendefinisikan data dan model proses yang tujuannya adalah pengembangan tradisional. UML menawarkan *Diagram* yang dikelompokkan manjadi lima perspektif berbeda untuk memodelkan suatu sistem. Seperti satu set *blue print* yang digunakan untuk membangun sebuah rumah (Saipul Anwar, et al., 2016 : 75-76).

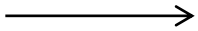
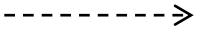
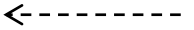
a. *Use case Diagram*

Use case Diagram merupakan pemodelan untuk kelakuakn (*behavior*) sistem in*form*asi yang akan dibuat. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem in*form*asi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *Use Case Diagram* yaitu:

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*.
	*Actor* atau 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>>	*Include*, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
 <<extends>>	*Extend*, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

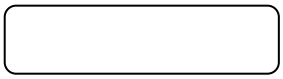
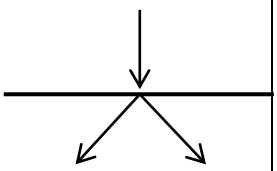
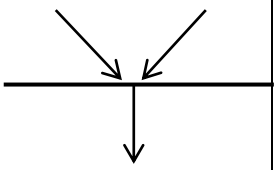
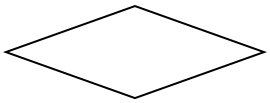
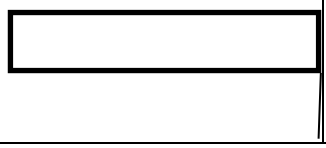
(Sumber : Ade Hendini, 2016: 108-109)

b. *Diagram* Aktivitas (A*ctivity 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.

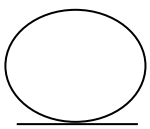
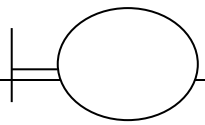
	*Activites*, mengGambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision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

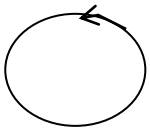
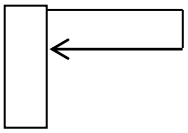
(Sumber : Ade Hendini, 2016: 109-110)

c. *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
	*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 *form*entry 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 : Ade Hendini, 2016: 110)

d. *Diagram* Kelas (*Class Diagram*)

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 *Assosiations*, *Generalitation* dan *Aggregation*, attribut (*Attributes*), operasi (*operation/method*) dan *visibility*, tingkat akses objek eksternal kepada suatu operasi atau attribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *Multiplicity* atau *Cardinality*.

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 : Ade Hendini, 2016: 1)