# BAB II
# TINJAUAN PUSTAKA

## II.1. Konsep Dasar

### II.1.1. Defenisi Sistem

Sistem adalah suatu kumpulan atau himpunan dari unsur, komponen, atau variabel yang terorganisir, saling berinteraksi, saling tergantung satu sama lain, dan terpadu. Teori sistem secara umum yang pertama kali diuraikan oleh Kenneth Boulding, terutama menekan pentingnya perhatian terhadap setiap bagian yang membentuk sebuah sistem.(*Sutabri; 2012*).

## II.2. Sistem Pakar

Sistem pakar (*Expert System*) merupakan solusi bagi AI (*Artificial Intelligence*) 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 : 3*).

### II.2.1. Karakteristik Sistem Pakar

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

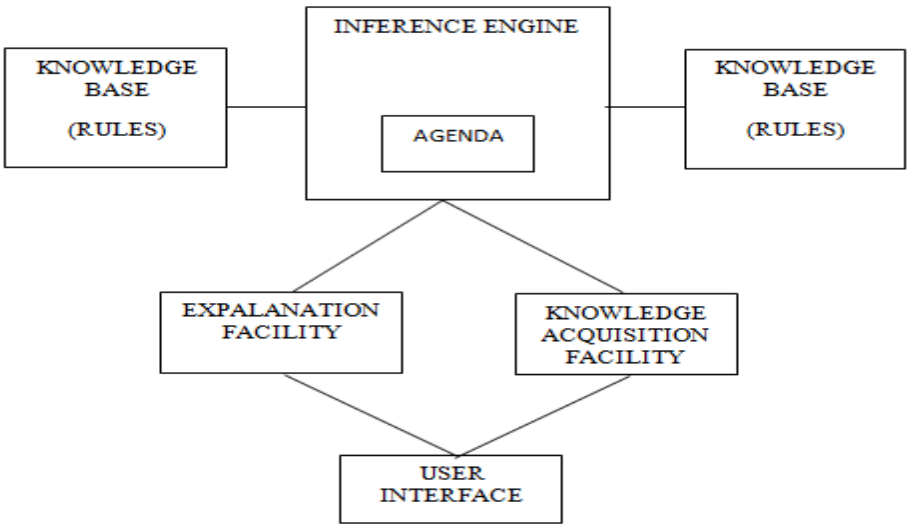
- Kinerja sangat baik (*high performance*). Sistem harus mampu memberikan respon berupa saran (*advice*) dengan tingkat kualitas yang sama dengan seorang pakar atau melebihinya.
- Waktu respon yang baik (*adequate respon time*). Sistem juga harus mampu bekerja dalam waktu yang sama baiknya (*reasonable*) atau lebih cepat dibandingkan dengan seorang pakar dalam menghasilkan keputusan. Hal ini sangat penting terutama pada sistem waktu nyata (*real-time*).

- Dapat diandalkan (*good reliability*). Sistem harus dapat diandalkan dan tidak mudah rusak/ *crash*.
- Dapat dipahami (*understandable*). Sistem harus mampu menjelaskan langkah-langkah penalaran yang dilakukannya seperti seorang pakar.

Fleksibel (*flexibility*). Sistem harus menyediakan mekanisme untuk menambah, mengubah, dan menghapus pengetahuan. (*Rika Rosnelly; 2012 ; 20*).

### II.2.2. Struktur Sistem Pakar

Pada Umumnya, antar muka pemakai juga berfungsi untuk menginputkan pengetahuan baru kedalam basis pengetahuan sistem pakar, menampilkan fasilitas penjelasan sistem dan memberikan tuntunan penggunaan sistem secara menyeluruh langkah demi langkah sehingga pemakai mengerti apa yang harus dilakukan terhadap sistem.




**Gambar II.1. Struktur Sistem Pakar**
**Sumber : (*Rosnelly; 2012 : 13*)**

Sistem pakar memiliki beberapa fitur menarik yang merupakan kelebihannya, seperti Meningkatnya ketersediaan (*increased availability*). Kepakaran atau keahlian menjadi tersedia dalam sistem komputer. Dapat dikatakan bahwa sistem pakar merupakan produksi kepakaran secara masal (*massproduction*).

## II.3. *Metode Certainty Factor*

Teori *Certainty Factor* (CF) diusulkan oleh Shortlife dan Buchanan pada 1975 untuk mengakomodasi ketidakpastian pemikiran (*inexact reasoning*) seorang pakar. Seorang pakar, (misalnya dokter) sering kali menganalisis informasi yang ada dengan ungkapan seperti "mungkin", "kemungkinan besar", hampir pasti". Untuk mangakomodasi hal ini kita menggunakan *certainty factor* (CF) guna menggambarkan tingkat keyakinan pakar terhadap masalah yang sedang dihadapi (*T.Sutojo, dkk , 2011 : 194*).

Ada dua cara dalam mendapatkan tingkat keyakinan (CF) dari sebuah rule yaitu (*T.Sutojo, dkk ; 2011 : 194-196*):

1. Metode *'Net Belief'* yang diusulkan oleh E. H. Shortliffe B. G. Buchanan

CF(Rule) = MB[H,E] – MD[H,E] .....(1)

$$\mathrm{MB}(\mathrm{H},\mathrm{E}) = \begin{cases} 1 & \mathrm{P}(\mathrm{H}) = 1 \\ \dfrac{\max[\mathrm{P}(\mathrm{H}|\mathrm{E}),\mathrm{P}(\mathrm{H})] - \mathrm{P}(\mathrm{H})}{\max[1,0] - \mathrm{P}(\mathrm{H})} & \text{lainnya} \end{cases} \dots\dots\dots\dots\dots\dots\dots\dots (2)$$

$$\mathrm{MD}(\mathrm{H},\mathrm{E}) = \begin{cases} 1 & \mathrm{P}(\mathrm{H}) = 0 \\ \dfrac{\min[\mathrm{P}(\mathrm{H}|\mathrm{E}),\mathrm{P}(\mathrm{H})] - \mathrm{P}(\mathrm{H})}{\min[1,0] - \mathrm{P}(\mathrm{H})} & \text{lainnya} \end{cases} \dots\dots\dots\dots\dots (3)$$

Dimana :

CF(Rule) = Faktor kepastian

MB(H,E) = *measure of belief*(ukuran kepercayaan) terhadap hipotesis H, jika diberikan *evidence*E (antara 0 dan 1)

MD(H,E)= *measure of disbelief*(ukuran ketidakpercayaan) terhadap *evidenc*eH, jika diberikan *evidence*E (antara 0 dan 1)

P(H) = Probabilitas kebenaran hipotesis H

P(H|E) = Probabilitas bahwa H benar karena fakta E

2. Dengan cara mewawancarai seorang pakar

Nilai CF (Rule) didapat dari interpretasi "term" dari pakar, yang diubah menjadi nilai CF tertentu sesuai tabel berikut.

**Tabel II.1. Nilai CF**

| **Uncertain Term** | **CF** |
|---|---|
| Defenitely not (pasti tidak) | -1.0 |
| Almost certainly not (hampir pasti tidak) | -0.8 |
| Probably not (kemungkinan besar tidak) | -0.6 |
| Maybe not (mungkin tidak) | -0.4 |
| Unknown (tidak tahu) | -0.2 to 0.2 |
| Maybe (mungkin) | 0.4 |
| Probably (kemungkinan besar) | 0.6 |
| Almost certainly (hampir pasti) | 0.8 |
| Definitely (pasti) | 1.0 |

**(*Sumber : T. Sutojo, dkk; 2011 : 195-196*)**

### II.3.1. Perhitungan *Certainty Factor* Gabungan

Secara umum, rule direpresentasikan dalam bentuk sebagai berikut (*T.Sutojo, dkk ;2011 : 196-198*) :

IF $E_1$ AND $E_2$ .....AND $E_n$ THEN H (CF Rule)

Atau

IF $E_1$ OR $E_2$ .....OR $E_n$ THEN H (CF Rule)

Di mana :

$E_1$ ... $E_n$ : Fakta-fakta (evidence) yang ada

H : Hipotesis atau konklusi yang dihasilkan

CF (Rule) : Tingkat keyakinan terjadinya hipotesis H akibat adanya fakta-fakta $E_1$ ... $E_n$

1. Rule dengan *evidence* E tunggal dan Hipotesis H tunggal

   IF E THEN H (CF rule)

   CF(H,E) = CF(E) x CF (rule).....(4)

   Catatan :

   Secara praktik, nilai CF rule ditentukan oleh pakar, sedangkan nilai CF(E) ditentukan oleh pengguna saat berkonsultasi dengan sistem pakar.

2. Rule dengan *evidence* E ganda dan Hipotesis H tunggal

   IF $E_1$ AND $E_2$ .....AND $E_n$ THEN H (CF Rule)

   $CF(H,E) = \min[CF(E_1), CF(E_2),....., CF(E_n)] \times CF(rule)$.....(5)

   IF $E_1$ OR $E_2$ .....OR $E_n$ THEN H (CF Rule)

   $CF(H,E) = \max[CF(E_1), CF(E_2),....., CF(E_n)] \times CF(rule)$.....(6)

3. Kombinasi dua buah rule dengan *evidence* berbeda ($E_1$ dan $E_2$), tetapi hipotesis sama.

   IF $E_1$ THEN H  Rule 1  $CF(H,E_1) = CF_1 = C(E_1)$ x CF (Rule1)

   IF $E_2$ THEN H  Rule 2  $CF(H,E_2) = CF_2 = C(E_2)$ x CF (Rule2)

$$CF(CF_1, CF_2) = \begin{cases} CF_1 + CF_2(1 - CF_1) \text{ jika } CF_1 > 0 \; dan CF_2 > 0 \\ \dfrac{CF_1 + CF_2}{1 - \min[|CF_1|, |CF_2|]} \text{ jika } CF_1 < 0 \; atau CF_2 < 0 \\ CF_1 + CF_2 \text{ x } (1 + CF_1) \text{ jika } CF_1 < 0 \; dan CF_2 < 0 \end{cases} \quad \ldots (7)$$

### II.3.2. Kelebihan dan Kekurangan Metode *Certainty Factor*

Kelebihan metode *Certainty Factor* adalah (*T.Sutojo, dkk ;2011 : 204*) :

1. Metode ini cocok dipakai dalam sistem pakar yang mengandung ketidakpastian.
2. Dalam sekali proses perhitungan hanya dapat mengolah 2 data saja sehingga kekurangan data dapat terjaga.

Sedangkan kekurangan metode *Certainty Factor* adalah:

1. Pemodelan ketidakpastian yang menggunakan perhitungan metode *Certainty Factor* biasanya masih diperdebatkan.
2. Untuk data lebih dari 2 buah, harus dilakukan beberapa kali pengolahan data.
   - Mengurangi biaya (*reduced cost*). Biaya yang diperlukan untuk menyediakan keahlian per satu orang *user* menjadi berkurang.
   - Mengurangi bahaya (*reduced danger*). Sistem pakar dapat digunakan di lingkungan yang mungkin berbahaya bagi manusia.
   - 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.

- 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.
- 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.
- 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.
- 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 relative memberikan respon yang lebih cepat dibandingkan seorang pakar.

- 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.
- 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;5*).

## II.4. Kualitas Kelapa Sawit

Kualitas adalah tingkat baik buruknya atau taraf atau derajat sesuatu (*Wikipedia*). Kelapa sawit (*Elaeis*) adalah tanaman perkebunan penting penghasil minyak makanan, minyak industri, maupun bahan bakar nabati (*biodiesel*). Pengelompokan kelapa sawit dapat dilakukan berdasarkan tebal tipisnya cangkang dan warnanya (*Risza,1994*).

1. Berdasarkan tebal tipisnya cangkang

Berdasarkan tebal tipisnya cangkang tanaman kelapa sawit dapat dibedakan menjadi 3 tipe yaitu:

a. Tipe Dura

Kelapa sawit memiliki tempurung (cangkang) yang sangat tebal, tetapi kandungan minyak dalam buahnya rendah.

b. Tipe Pisifera

Kelapa sawit memeliki tempurung (cangkang) yang sangat tipis bahkan hanya berbentuk banyangan cincin, namun kandungan minyak dalam buah tinggi.

c. Tipe Tenera

Merupahkan persilangan antara dura sebagai pohon ibu, dengan pisifera sebagai pohon bapak.Tenera memiliki tempurung yang tipis dan kandungan minyak tinggi.

2. Berdasarkan Warna Buah

Berdasarkan warna buah tanaman kelapa sawit dapat dibedakan menjadi 3 bentuk yaitu :

a. Nigrescens

Warna buahnya Lembayung (Violet) sampai hitam pada waktu masih muda,kemudian menjadi warna merah kuning (Orange) sesudah matang.

b. Virescens

Warna buahnya hijau pada waktu muda, kemudian berubah menjadi merah kekuningan sesudah matang.

c. Albescens

Warna Buahnya kuning pada waktu muda dan pucat serta tembus cahaya kerena mengandung sedikit koraten.

## II.5. Basis Data

Basis data dapat dipahami sebagai suatu kumpulan data terhubung (*interrelated data*) yang disimpan secara bersama-sama pada suatu media, tanpa *mengatap* satu sama lain atau tidak perlu suatu kerangkapan data (kalaupun ada maka kerangkapan data tersebut harus seminimal mungkin dan terkontrol [*controlled redudancy*] ), data disimpan dengan cara-cara tertentu sehingga mudah digunakan/atau ditampilkan kembali; data dapat digunakan oleh satu atau lebih program-program aplikasi secara optimal; data disimpan tanpa mengalami ketergantungan dengan program yang akan menggunakannya; data disimpan sedemikian rupa sehingga proses penambahan, pengambilan, dan modifikasi data dapat dilakukan dengan mudah dan terkontrol (*Sutanta, 2011 : 29-30*).

## II.6 *Unified Modelling Language* (UML)

*Unified Modeling Language* (UML) adalah sebuah bahasa yang diterima dan digunakan oleh *software developer* dan *software analyst* sebagai suatu bahasa yang cocok untuk merepresentasikan grafik dari suatu relasi antar entitas-entitas software (*Gornik, 2003*). Dengan menggunakan UML, tim pengembang *software*akan mempunyai banyak keuntungan, seperti memudahkan komunikasi dengan sesama anggota tim tentang *software* apa yang akan dibuat, memudahkan integrasi ke dalam area pengerjaan *software* karena bahasa ini berbasiskan *meta-*

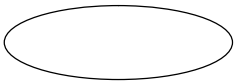
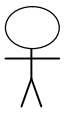
*models* dimana *meta-models* bisa mendefinisikan proses-proses untuk mengkonstruksikan konsep-konsep yang ada. UML juga menggunakan formatur *input* dan *output* yang sudah mempunyai bentuk standar yaitu *XML Metadata Interchange* (XMI), menggunakan aplikasi dan pemodelan data yang universal, merepresentasikan dari tahap analisis ke implementasi lalu ke *deployment* yang terpadu, dan mendeskripsikan keutuhan tentang spesifikasi software.

UML menyediakan kumpulan alat yang sudah terstandarisasi, yang digunakan untuk mendokumentasikan analisis dan perancangan sebuah sistem perangkat lunak. (*Kendall & Kendall, 2005, p663*) Peralatan utama UML adalah diagram-diagram yang digunakan untuk membantu manusia dalam memvisualisasikan proses pengembangan sebuah sistem perangkat lunak, sama seperti penggunaan denah (*blueprint*) dalam pembuatan bangunan (*Edgar Winata dan Johan Setiawan, 2013*).

### II.6.1. *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.2 dibawah ini :

**Tabel II.2. Simbol *Use Case***

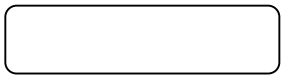
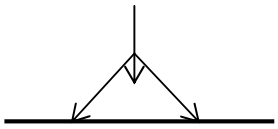
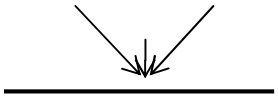
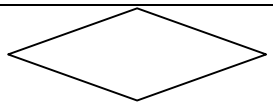
| Gambar | Keterangan |
|---|---|
|  | *Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*. |
|  | Aktor adalah *abstraction* dari orang atau sistem yang 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 : 94*)**

### II.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* dapat dilihat pada tabel II.2 dibawah ini :

**Tabel II.3. Simbol *Activity Diagram***

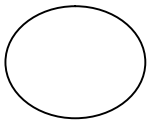
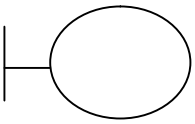
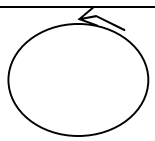
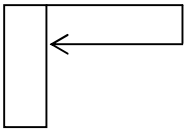
| Gambar | Keterangan |
|---|---|
|  | *Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas. |
|  | *End point*, akhir aktifitas. |
|  | *Activites*, menggambarkan suatu proses/kegiatan bisnis. |
|  | *Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara 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 : 94***)

### II.4.3. 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.4 dibawah ini :

**Tabel II.4. 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 : 95*)**

### II.4.4. *Class* Diagram

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.5 dibawah ini :

**Tabel II.5. *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 : 95)**

## II.5. *PHP*

PHP singkatan dari *Hypertext Preprocessor* Yaitu bahasa pemrograman web server side yang bersifat open source. PHP merupakan script yang terintegrasi dengan HTML dan berada pada server ( *server side HTML embedded scripting*). PHP dalah script yang digunakan untuk membuat halaman website yang dinamis. Dinamis berarti halaman yang akan ditampilkan dibuat saat halaman itu diminta oleh client. Mekanisme ini menyebabkan informasi yang diterima client selalu yang terbaru/*up to date*.Semua script PHP dieksekusi pada server dimana script tersebut dijalankan.(*Anhar; 2010 : 3*).

Beberapa keunggulan yang dimiliki Program PHP adalah :

a. PHP memiliki tingkat akses yang lebih cepat.
b. PHP memiliki tingkat Lifecycle yang cepat sehingga selalu mengikuti perkembagan teknologi internet.
c. PHP memiliki tingkat keamanan yang tinggi.
d. PHP mampu berjalan di beberapa server yang ada, misalnya Apache, Miscrosoft IIS, PWS, AOLserver, phttpd, fhttpd dan Xitami
e. PHP mampu berjalan di Linux sebagai platform sebagai sistem operasi utama bagi PHP, namun juga dapat berjalan di FreeBSD, Unix, Solaris, Windows, dan yang lain.
f. PHP juga mendukung akses ke beberapa database yang sudah ada, baik yang bersifat free/gratis ataupun komersial. Database itu antara lain MySQL, PosgreSQL, mSQL, Informix, dan MicrosoftSQL server.PHP bersifat free dan gratis.(*Andi Offset, 2004; 2*)

**II.8. *MYSQL***

MYSQL (*my Structure Query Language*) adalah sebuah perangkat lunak system manajemen basis data SQL(*database management system*) atau DBMS dari sekian banyak DBMS seperti, *Oracle*, *MS SQL*, *Postagre SQL*, dan lain-lain. *MYSQL*merupakan DBMS yang multithread, yaitu multi user tidak seperti apache yang merupakan software yang dikembangkan oleh komunitas umum, dan hak cipta untuk kode user dimiliki oleh penulisnya masing-masing. *MYSQL* dimilki dan disponsori oleh sebuah perusahaan swedia yaitu, *MYSQL AB* memegang hak cipta mendirikan*MYSQL AB* adalah : *David Axmark*, *Allan Larson* dan *Michael*

*Monty Widenius*. Seperti yang telah disebutkan sebelumnya*MYSQL* bersifat gratis atau open source sehingga kita bisa menggunakannya secara gratis. Pemrograman PHP juga sangatmendukung /*support* dengan sungguh-sungguh kita dapat mengaplikasikan PHP dan *MYSQL*dalam membuat aplikasi website dalam membuat website.(*Anhar , 2010 ; 21*)