BAB II

TINJAUAN PUSTAKA

II.1. Pengertian Sistem

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

1. Komponen Sistem

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

2. Batasan Sistem

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

3. Lingkungan Luar Sistem

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

4. Penghubung Sistem

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

5. Masukan Sistem

   Masukan sistem adalah energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan (*maintance input*) dan masukan sinyal (*signal input*).

6. Keluaran Sistem

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

7. Pengolah Sistem

   Suatu sistem dapat mempunyai suatu bagian pengolah atau sistem itu sendiri sebagai pengolahnya. Pengolah akan mengubah masukan menjadi keluaran.

8. Sasaran Sistem

   Suatu sistem mempunyai tujuan (*goal*) atau sasaran (*objective*). Kalau suatu sistem tidak mempunyai sasaran, maka operasi sistem tidak ada gunanya (Sulindawati, 2010 : 135).

II.2. Sistem Pakar

Sistem pakar menirukan perilaku seorang pakar dalam menangani suatu persoalan. Pada suatu kasus seorang pasien mendatangi dokter untuk memeriksa

badannya yang mengalami gangguan kesehatan, maka dokter atau pakar kesehatan akan memeriksa dan melakukan diagnosa. Bila dokter cukup sibuk dan

Pelaksana diagnosa digantikan oleh sebuah sistem pakar, maka sistem pakar diharapkan dapat membantu memahami dan menganalisa keadaan pasien dan menemukan penyakit yang diderita pasien itu. Sistem pakar diharapkan juga untuk menghasilkan dugaan atau hasil diagnosa yang sama dengan diagnosa yang dilakukan oleh seorang ahli. Tujuan utama sistem pakar bukan untuk menggantikan kedudukan seorang ahli maupun pakar, tetapi untuk memasyarakatkan pengetahuan dan pengalaman pakar-pakar yang ahli di bidangnya (Andri Saputra, 2011 : 203).

II.2.1. Konsep Sistem Pakar

Konsep dasar sistem pakar mengandung : keahlian, ahli, pengalihan keahlian, inferensi, aturan dan kemampuan menjelaskan. Keahlian adalah suatu kelebihan penguasaan pengetahuan di bidang tertentu yang diperoleh dari pelatihan, membaca atau pengalaman. Contoh bentuk pengetahuan yang termasuk keahlian adalah :

1. Fakta-fakta pada lingkup permasalahan tertentu.
2. Teori-teori pada lingkup permasalahan tertentu.
3. Prosedur-prosedur dan aturan-aturan berkenaan dengan lingkup permasalahan tertentu.
4. Strategi-strategi global untuk menyelesaikan masalah.
5. Meta-knowledge (pengetahuan tentang pengetahuan).

Bentuk-bentuk ini memungkinkan para ahli untuk dapat mengambil keputusa lebih cepat dan lebih baik daripada seseorang yang bukan ahli. Seorang ahli adalah seseorang yang mampu menjelaskan suatu tanggapan, mempelajari hal-hal baru seputar topik permasalahan (domain), menyusun kembali pengetahuan jika dipandang perlu, memecah aturan-aturan jika dibutuhkan, dan menentukan relevan tidaknya keahlian mereka. Pengalihan keahlian dari para ahli ke komputer untuk kemudian dialihkan lagi ke orang lain yang bukan ahli, merupakan tujuan utama dari sistem pakar. Proses ini membutuhkan 4 aktivitas yaitu :

1. Tambahan pengetahuan (dari para ahli atau sumber-sumber lainnya).
2. Representasi pengetahuan (ke komputer).
3. Inferensi pengetahuan.
4. Pengalihan pengetahuan ke user.

II.2.2. Basis Pengetahuan

Pengetahuan yang disimpan di komputer disebut dengan nama basis pengetahuan. Ada 2 tipe pengetahuan, yaitu : fakta dan prosedur (biasanya berupa aturan). Salah satu fitur yang harus dimiliki oleh sistem pakar adalah kemampuan untuk menalar. Jika keahlian-keahlian sudah tersimpan sebagai basis pengetahuan dan sudah tersedia program yang mampu mengakses basisdata, maka komputer harus dapat diprogram untuk membuat inferensi. Proses inferensi ini dikemas dalam bentuk motor inferensi (inference engine). Sebagian besar sistem pakar komersial dibuat dalam bentuk rule-based systems, yang mana pengetahuannya

disimpan dalam bentuk aturan-aturan. Aturan tersebut biasanya berbentuk IF-THEN. Fitur lainnya dari sistem pakar adalah kemampuan untuk merekomendasi. Kemampuan inilah yang membedakan sistem pakar dengan sistem konvensional. Adapun 4 bentuk sistem pakar, yaitu :

1. Berdiri sendiri

   Sistem pakar jenis ini merupakan software yang berdiri-sendiri tidak tergantung dengan software yang lainnya.

2. Tergabung

   Sistem pakar jenis ini merupakan bagian program yang terkandungdidalam suatu algoritma (konvensional), atau merupakan program dimana didalamnya memanggil algoritma subrutin lain (konvensional).

3. Menghubungkan ke software lain

   Bentuk ini biasanya merupakan sistem pakar yang menghubungkan ke suatu paket program tertentu, misalnya DBMS.

4. Sistem Mengabdi

   Sistem pakar merupakan bagian dari komputer khusus yang dihubungkan dengan suatu fungsi tertentu. Misalnya sistem pakar yang digunakan untuk membantu menganalisis data radar.

II.2.3. Lingkungan Sistem Pakar

Sistem pakar teridiri-dari 2 bagian pokok, yaitu : lingkungan pengembangan (*development environment*) dan lingkungan konsultasi

(*consultation environment*). Lingkungan pengembangan digunakan sebagai pembangunan sistem pakar baik dari segi pembangunan komponen maupun basis pengetahuan. Lingkungan konsultasi digunakan oleh seorang yang bukan ahli untuk berkonsultasi.

Sistem Konvensional Informasi dan pemrosesannya biasanya jadi satu dengan program Biasanya tidak bisa menjelaskan mengapa suatu input data itu dibutuhkan, atau bagaimana output itu diperoleh Pengubahan program cukup sulit & membosankan Sistem hanya akan beroperasi jika sistem tersebut sudah lengkap Eksekusi dilakukan langkah demi langkah Menggunakan data Tujuan utamanya adalah efisiensi Sistem Pakar Basis pengetahuan merupakan bagian dari mekanisme inferensi Penjelasan adalah bagian terpenting dari sistem pakar Pengubahan aturan dapat dilaksanakan dengan mudah Sistem dapat beroperasi hanya dengan beberapa aturan. Eksekusi dilakukan pada keseluruhan basis pengetahuan Menggunakan pengetahuan Tujuan utamanya adalah efektivitas (Ari Fadli, 2010 : 2).

II.3. Metode *Hebb Rule*

Hebb rule adalah aturan pelatihan yang paling awal dan paling sederhana untuk jaringan syaraf tiruan secara umum. Pada aturan hebb ini pelatihan yang terjadi yaitu dengan memodifikasi kekuatan sinapsis (bobot). Jika kedua syaraf kedua-duanya “on” pada waktu yang sama, maka bobot neuron akan bertambah.

Kita akan mengarah pada jaringan syaraf single layer (*feedforward*) dilatih menggunakan (perluasan hebb rule sebagai jaringan hebb. Aturan hebb juga digunakan untuk pelatihan jaringan lain yang dibahas kemudian. Karena kita

sedang mempertimbangkan jaringan single-layer, salah satu neuron yang saling berhubungan merupakan satu unit masukan dan satu unit keluaran (karena tidak ada unit input yang terhubung satu sama lain, tidak juga banyak unit output terhubung). Jika data ditunjukkan dalam bentuk bipolar, ini mudah untuk menyatakan pembaharuan bobot yang diinginkan sehingga :

$$W_i\ (new) = W_i\ (old) + x_i\ y$$

Jika data adalah biner, formula ini tidak membedakan antara pasangan pelatihan di mana unit input adalah “on” dan nilai target adalah “off” dan pasangan pelatihan yang mana antara unit input dan nilai target adalah “off” (Yun Eninggar ; 2012 : 2).

II.3.1. Langkah – langkah Perhitungan Metode Hebb Rule

Misalkan kita menggunakan pasangan *vektor input* (s) dan *vektor output* sebagai pasangan *vektor* yang akan dilatih. Sedangkan *vektor* yang hendak digunakan sebagai testing adalah *vektor* (x).

Algoritma pasangan *vektor* diatas adalah sebagai berikut:

1. Insialisasi semua bobot dimana:

   $w_{ij}=0$;

   i =1,2……n;

   j =1,2…..m

2. Pada pasangan input-output(s-t), maka terlebih dahulu dilakukan langkah-langkah sebagai berikut :

   a. Set input dengan nilai yang sama dengan *vektor* inputnya:

$x_i=s_i$; (i=1,2……,n)

b. Set outputnya dengan nilai yang sama dengan *vektor* outputnya:

$y_j=t_j$; (j=1,2,…..,m)

c. Kemudian jika terjadi kesalahan maka perbaiki bobotnya:

w_{ij}(baru)=w_{ij}(lama)+x_i*y_j;

(i=1,2,….,n dan j=1,2,….,m)

sebagai catatan bahwa nilai bias selalu 1.

Misalkan kita gunakan pasangan *vektor* input (s) dan *vektor* output (t) sebagai pasangan *vektor* yang akan dilatih.

Algoritmanya adalah :

1. Inisialisasi semua bobot dan bias:

   (agar penyelesaiannya jadi lebih sederhana maka kita set semua bobot dan bias sama dengan nol).

   Set learning rate( ): (0< 1).

2. Selama kondisi berhenti bernilai false. Lakukan langkah-langkah sebagai berikut:

   (i) Untuk setiap pasangan pembelajaran s-t, maka:

   a. Set input dengan nilai sama dengan *vektor* input:

      $x_i=s_i$;

   b. Hitung respon untuk unit output:

      y_{in}=b+y-

   c. Perbaiki bobot dan bias jika terjadi error:

      jika y≠t maka:

w_i(baru)=w_i(lama)+

b(baru)=b(lama)+

jika tidak, maka:

w_i(baru)= w_i(lama)

b(baru)=b(lama)

Selanjutnya lakukan tes kondisi berhenti: jika tidak terjadi perubahan pada bobot pada (i) maka kondisi berhenti adalah *TRUE*, namun jika masih terjadi perubahan pada kondisi berhenti maka *FALSE* (Muhammad Rofiq ; 2013 : 2).

II.4. Pengertian PHP

PHP merupakan suatu bahasa pemrograman sisi server yang dapat anda gunakan untuk membuat halaman Web dinamis. Contoh bahasa yang lain adalah *Microsoft Active Server Page (*ASP) dan *Java Server Page* (JSP). Dalam suatu halaman HTML anda dapat menanamkan kode PHP yang akan dieksekusi setiap kali halaman tersebut dikunjungi. Karena kekayaannya akan fitur yang mempermudah perancangan dan pemrograman Web, PHP memiliki pepularitas yang tinggi. Anda dapat mengecek *survey* popularitas yang dilakukan *netcraft* di URL www.php.net/usage.php.

PHP adalah kependekan dari *HyperText Prepocessor* (suatu akronim rekursif) yang dibangun oleh Rasmus Lerdorf pada tahun 1994. Dahulu, pada awal pengembangnnya PHP disebut sebagai kependekan dari *Personal Home Page.* PHP merupakan produk *Open Source* sehingga anda dapat mengakses

source code, menggunakan dan mengubahnya tanpa harus membayar sepeser pun. Gratis (Antonius Nugraha Widhi Pratama ; 2010 : 9).

II.5. Pengertian MySQL

Mysql pertama kali dirintis oleh seorang programmer database bernama Michael Widenius, yang dapat anda hubungi di emailnya monty@analytikerna. Mysql *database server* adalah RDBMS (*Relasional Database Management System*) yang dapat menangani data yang bervolume besar. meskipun begitu, tidak menuntut *resource* yang besar. Mysql adalah *database* yang paling populer di antara *database* yang lain.

MySQL adalah program *database* yang mampu mengirim dan menerima data dengan sangat cepat dan *multi user*. MySQL memiliki dua bentuk lisensi, yaitu *free software* dan *shareware*. penulis sendiri dalam menjelaskan buku ini menggunakan *database* ini untuk keperluan pribadi atau usaha tanpa harus membeli atau membayar lisensi, yang berada di bawah lisensi GNU/GPL (*general public license*) (Wahana Komputer; 2010 : 5).

II.6. Teknik Normalisasi

Normalisasi adalah teknik perancangan yang banyak digunakan sebagai pemandu dalam merancang basis data relasional. Pada dasarnya, normalisasi adalah proses dua langkah yang meletakkan data dalam bentuk tabulasi dengan menghilangkan kelompok berulang lalu menghilangkan data yang terduplikasi dari tabel rasional.

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

1. Bentuk normal tahap pertama (1” Normal Form)

Contoh yang kita gunakan di sini adalah sebuah perusahaan yang mendapatkan barang dari sejumlah pemasok. Masing-masing pemasok berada pada satu kota. Sebuah kota dapat mempunyai lebih dari satu pemasok dan masing-masing kota mempunyai kode status tersendiri. Contoh normalisasi 1NF adalah seperti pada table II.1 :

Tabel II.1. Tabel Bentuk Normal Pertama (1NF)

p#	status	kota	b#	qty
p1	20	Yogyakarta	b1	300
p1	20	Yogyakarta	b2	200
p1	20	Yogyakarta	b3	400
p1	20	Yogyakarta	b4	200
p1	20	Yogyakarta	b5	100
p1	20	Yogyakarta	b6	100
p2	10	Medan	b1	300
p2	10	Medan	b2	400
p3	10	Medan	b2	200
p4	20	Yogyakarta	b2	200
p4	20	Yogyakarta	b4	300
p4	20	Yogyakarta	b5	400

2. Bentuk normal tahap kedua (2nd normal form)

Definisi bentuk normal kedua menyatakan bahwa tabel dengan kunci utama gabungan hanya dapat berada pada 1NF, tetapi tidak pada 2NF. Sebuah tabel relasional berada pada bentuk normal kedua jika dia berada

pada bentuk normal kedua jika dia berada pada 1NF dan setiap kolom bukan kunci yang sepenuhnya tergantung pada seluruh kolom yang membentuk kunci utama pada table II.2.

Tabel II.2. Tabel Bentuk Normal Kedua (2NF)

Pemasok2

p#	Status	Kota
P1	20	Yogyakarta
P2	10	Medan
P3	10	Medan
P4	20	Yogyakarta
P5	30	Bandung

Barang

p#	b#	qty
p1	b1	300
p1	b2	200
p1	b3	400
p1	b4	200
p1	b5	100
p1	b6	100
p2	b1	300
p2	b2	400
p3	b2	200
p4	b2	200
p4	b4	300
p4	b5	400

3. Bentuk normal tahap ketiga (3rd normal form)

Bentuk normal ketiga mengharuskan semua kolom pada tabel relasional tergantung hanya pada kunci utama. Secara definisi, sebuah tabel berada pada bentuk normal ketiga (3NF) jika tabel sudah berada pada 2NF dan setiap kolom yang bukan kunci tidak tergantung secara transitif pada kunci utamanya pada table II.3.

Tabel II.3. Tabel Bentuk Normal Ketiga (3NF)

Pemasok Kota

p#	Kota
P1	Yogyakarta
P2	Medan
P3	Medan
P4	Yogyakarta
P5	Bandung

Kota Status

Kota	status
Yogyakarta	20
Medan	10
Yogyakarta	20
Bandung	30

4. *Boyce Code Normal Form* (BCNF)

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

5. Bentuk Normal Keempat (4NF)

Sebuah tabel rasional berada pada bentuk normal keempat (4NF) jika dia dalam BCNF dan semua ketergantungan multivalue merupakan ketergantungan fungsional. Bentuk normal keempat (4NF) didasarkan pada konsep ketergantungan multivalue (MVD). Sebuah ketergantungan multivalue tiga kolom, satu kolom mempunyai banyak baris bernilai sama, tetapi kolom lain bernilai berbeda.

Tabel II.4. Tabel Bentuk Normal Keempat (4NF)

Pegawai Proyek

peg#	Pry#
1211	P1
1211	P3

Pegawai Ahli

Peg#	ahli
1211	Analisis
1211	Perancangan
1211	Pemrograman

4. Bentuk Normal Kelima

Sebuah tabel berada pada bentuk normal kelima (5NF) jika ia tidak dapat mempunyai dekomposisi lossless menjadi sejumlah tabel lebih kecil. Empat bentuk normal pertama berdasarkan pada konsep ketergantungan fungsional,

sedangkan bentuk normal kelima berdasarkan pada konsep ketergantungan gabungan (*join dependence*) (Janner Simarmata ; 2010 : 78).

Tabel II.5. Tabel Bentuk Normal Kelima (5NF)

peg#	Pry#	Ahli
1211	11	Perancangan
1211	28	Pemrograman

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

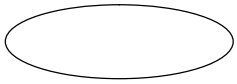
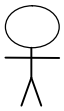
Menurut Windu Gata (2013 : 4) Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *Unified Modeling Language* (UML). UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak. UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem. Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

1. *Use case* Diagram

   Use case diagram merupakan pemodelan untuk melakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di

dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram, yaitu :

Tabel II.6. Simbol *Use Case*

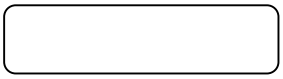
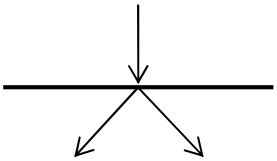
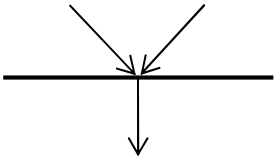
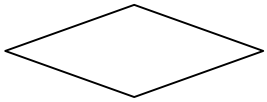
Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi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

(Sumber : Windu Gata, 2013 : 4)

2. Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram,* yaitu :

Tabel II.7. Simbol *Activity Diagram*

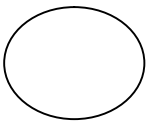
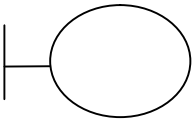
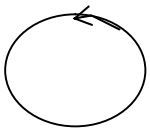
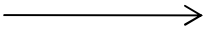
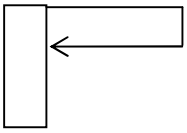
Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan 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 : Windu Gata, 2013 : 6)

3. Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram*, yaitu :

Tabel II.8. 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 : Windu Gata, 2013 : 7)

4. *Class Diagram* (Diagram Kelas)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggng jawab entitas yang menentukan perilaku sistem. *Class diagram* juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*), *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti.

Tabel II.9. *Multiplicity Class Diagram*

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

(Sumber : Windu Gata, 2013 : 9)