BAB II

TINJAUAN PUSTAKA

II.1. Sistem

Sistem merupakan kumpulan dari unsur atau elemen-elemen yang saling berkaitan/berinteraksi dan saling memengaruhi dalam melakukan kegiatan bersama untuk mencapai suatu tujuan tertentu. Menurut Jerry FithGerald, Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan dan berkumpul bersama-sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sasaran tertentu (Asbon Hendra ; 2012 : 157).

Berdasarkan prinsip dasar secara umum, sistem terbagi dalam:

1. Sistem *Terspesialisasi* adalah sistem yang sulit diterapkan pada lingkungan yang berbeda, misalnya sistem biologi: ikan yang dipindahkan ke darat.
2. Sistem Besar adalah sistem yang sebagian besar sumber dayanya berfungsi melakukan perawatan harian. Misalnya dinosaurus sebagai sistem biologi menghabiskan sebagian besar masa hidupnya dengan makan.
3. Sistem Sebagai Bagian dari Sistem Lain. Sistem selalu merupakan bagian dari sistem yang lebih besar, dan dapat terbagi menjadi sistem yang lebih kecil.
4. Sistem Berkembang. Walaupun tidak berlaku bagi semua sistem, hampir semua sistem selalu berkembang (Asbon Hendra ; 2012 : 163).

II.1.1. Karakteristik Sistem

1. Komponen (*Component*)

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

2. Batas Sistem (*Boundary*).

   Batas sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batas sistem ini memungkinkan suatu sistem dipandang sebagai suatu kesatuan, karena dengan batas sistem ini fungsi dan tugas dari subsistem yang satu dengan lainnya berbeda tetapi tetap saling berintekrasi. Batas suatu sistem menunjukkan ruang lingkup (*scope*) dari sistem tersebut.

3. Lingkungan Luar Sistem (*Environment*)

   Environment merupakan segala sesuatu di luar batas sistem yang memengaruhi operasi dari suatu sistem. Lingkungan Luar Sistem ini dapat bersifat menguntungkan atau merugikan. Lingkungan luar yang menguntungkan harus dipelihara dan dijaga agar tidak hilang pengaruhnya, sedangkan lingkungan luar yang merugikan harus dimusnahkan atau dikendalikan agar tidak mengganggu operasi sistem.

4. Penghubung Sistem (*Interface*)

   Merupakan media penghubung antara satu subsistem dengan subsistem yang lainnya untuk membentuk satu kesatuan sehingga sumber-sumber daya

mengalir dari subsistem yang lainnya. Dengan kata lain, *output* dari suatu subsistem akan menjadi *input* dari subsistem yang lainnya.

5. Masukan Sistem (*Input*)

   Merupakan energi yang dimasukkan ke dalam sistem. Masukan dapat berupa Masukan Perawatan (*Maintenance Input*) adalah energi yang dimasukkan supaya sistem tersebut dapat beroperasi. Masukan sinyal (*Signal Input*) adalah energi yang diproses untuk didapatkan keluaran.

6. Keluaran Sistem (*Output*)

   Merupakan hasil dari energi yang diolah oleh sistem, meliputi *output* yang berguna, contohnya informasi yang dikeluarkan oleh komputer. Dan *output* yang tidak berguna dikenal sebagai sisa pembangunan contohnya panas yang dikeluarkan oleh komputer.

7. Pengolah Sistem (*Process*)

   Merupakan bagian yang memproses masukan untuk menjadi keluaran yang diinginkan. Contoh CPU pada komputer, bagian produksi yang mengubah bahan baku menjadi barang jadi, serta bagian akuntansi yang mengolah data trannsaksi menjadi laporan keuangan.

8. Tujuan Sistem (*Goal*)

   Setiap sistem pasti mempunyai tujuan ataupun sasaran yang memengaruhi *input* yang dibutuhkan dan *output* yang dihasilkan. Dengan kata lain, suatu sistem akan dikatakan berhasil kalau pengoperasian sistem itu mengenai sasaran atau tujuannya. Jika sistem tidak mempunyai sasaran, maka operasi sistem tidak akan ada gunanya (Asbon Hendra ; 2012 : 158-160).

II.2. Sistem Pakar

Menurut 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 sehingga membutuhkan keahlian khusus dari manusia. Dengan kata lain, Sistem Pakar adalah sistem komputer yang ditujukan untuk meniru semua aspek (*emulates*) kemampuan pengambilan keputusan (*decision making*) seorang pakar. Sistem pakar memanfaatkan secara maksimal pengetahuan khusus selayaknya seorang pakar untuk memecahkan masalah (Rika Rosnelly ; 2012 : 2).

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

Seorang pakar memiliki kemampuan kepakaran, yaitu (Rika Rosnelly; 2012 : 10) :

1. Dapat mengenali dan merumuskan suatu masalah.
2. Menyelesaikan masalah dengan cepat dan tepat.
3. Menjelaskan solusi dari suatu masalah.
4. Restrukturisasi pengetahuan.
5. Belajar dari pengalaman.
6. Memahami batas kemampuan.

II.2.1. Kelebihan Sistem Pakar

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

1. Meningkatkan 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 multipel (*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 kepercayan 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 stress.

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 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 *relative* 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.
11. Basis data cerdas (*intelligent database*). Sistem pakar dapat digunakan untuk mengakses basis data secara cerdas (Rika Rosnelly ; 2012 : 5-6).

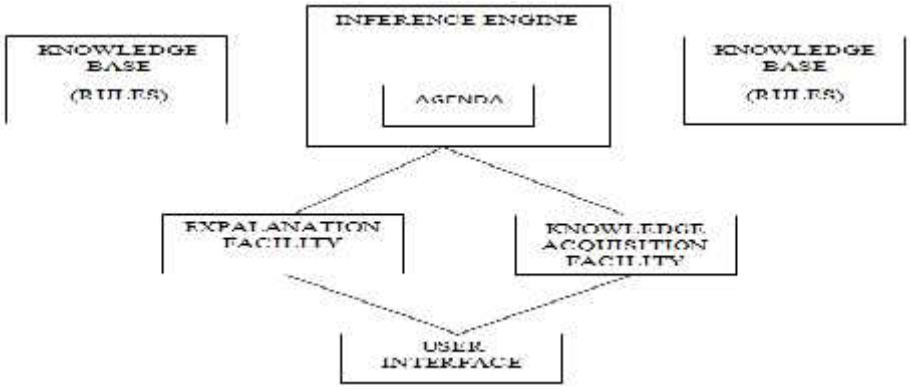
II.2.2. Ciri-Ciri Sistem Pakar

Ciri-ciri dari sistem pakar adalah sebagai berikut (T. Sutojo dkk, 2011 : 162) :

1. Terbatas pada dominan keahlian tertentu.
2. Dapat memberikan penalaran untuk data-data yang tidak lengkap atau tidak pasti.
3. Dapat menjelaskan alasan-alasan dengan cara yang dapat dipahami.
4. Bekerja berdasarkan kaidah/ rule tertentu.
5. Mudah dimodifikasi.
6. Basis pengetahuan dan mekanisme inferensi terpisah.
7. Keluarannya bersifat anjuran.
8. Sistem dapat mengaktifkan kaidah secara sesuai, dituntun oleh dialog dengan pengguna.

II.2.3. Struktur Sistem Pakar

Adapun struktur sistem pakar dapat dilihat pada gambar II.1.



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

Komponen yang terdapat dalam struktur sistem pakar ini adalah (Rika Rosnelly : 2012 : 14) :

1. *Knowledge Base* (Basis Pengetahuan)

   Basis pengetahuan mengandung pengetahuan untuk pemahaman, formulasi, dan penyelesaian masalah. Komponen sistem pakar disusun atas dua elemen dasar, yaitu fakta dan ukuran.

2. *Inference Engine* (Mesin Inferensi)

   Mesin Inferensi merupakan otak dari sebuah sistem pakar dan dikenal juga dengan sebutan control structure (struktur kontrol) atau *rule interpreter* (dalam sistem pakar berbasis kaidah).

3. *Working Memory*

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

4. *Explanation Facility*

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

5. *Knowledge Acquisition Facility*

   Meliputi proses pengumpulan, pemindahan dan perubahan dari kemampuan pemecahan masalah seorang pakar atau sumber pengetahuan terdokumentasi ke program komputer, 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 kedalam bentuk yang dimengerti oleh pemakai.

II.3. Penyakit *Askariasis*

Cacing *Ascaris Lumbricoides* dewasa berbentuk bulat dan besar, panjangnya dapat mencapai 15 - 30 cm. Sehingga akan menempati ruang yang luas dalam rongga usus. Bagi Anak yang mengandung cacing gelang dengan jumlah 300 ekor tidak akan merasa lapar; keadaan ini tentunya akan mengurangi masukan makanan bagi anak. Jumlah cacing yang banyak sangat berhubungan dengan terjadinya malnutrisi, defisit pertumbuhan dan gangguan kebugaran fisik, disamping itu massa cacing itu sendiri dapat menyebabkan *obstruksi*.

Cacing ini dapat hidup dalam tubuh pasien selama 12-18 bulan. Hidup dalam rongga usus halus manusia mengambil makanan terutama karbohidrat dan protein, seekor cacing akan mengambil karbohidrat 0.14 gram per hari dan protein 0.035 gram per hari. Sel mukosa usus halus (*enterosit*) mempunyai brush border yang terdiri dari mikrovili. Didalam mikrovili ini terdapat berbagai macam enzim pencernaan. Adanya *Ascaris Lumbricoides* dalam usus halus dapat menyebabkan kelainan mukosa usus, berupa proses peradangan pada dinding usus, pelebaran dan memendeknya *villi*, bertambah panjangnya kripta, menurunnya rasio *villus*

kripta dan *infiltrasi* sel bulat ke *lamina propria*, yang berakibat pada gangguan absorpsi makanan. Sebagian kelainan ini dapat kembali normal bila cacing dikeluarkan. Efek langsung yang terukur akibat kelainan mukosa usus halus ialah meningkatnya nitrogen dalam tinja, *steatorhea* karena terjadi gangguan *absorpsi* lemak, gangguan *absorbsi* karbohidrat yang diukur dengan *xylose* test. Akibat lainnya adalah cacing ini menyebabkan *hiperperistaltik* sehingga menimbulkan diare, juga dapat mengakibatkan rasa tidak enak diperut, *kolik* akut pada daerah *epigastrium* dan gangguan selera makan. Keadaan ini terjadi pada saat proses peradangan pada dinding usus.

Gangguan *absorpsi* vitamin A dapat terjadi pada anak yang menderita *askariasis*. Kekurangan vitamin A dapat menghambat pertumbuhan sel, termasuk sel tulang, dan dapat mengganggu sel yang membentuk *email* serta dentin pada gigi. Namun sampai dimana vitamin A digunakan dan dihancurkan oleh cacing masih belum jelas, namun dampak klinis defisiensi vitamin tersebut pada penderita *askariasis* dapat lebih berat. Juga didapati bukti penting mengenai efek terbalik *askariasis* terhadap vitamin *C* dalam plasma dan sekresi *riboflavin* dalam air seni (Charles D Siregar ; 113 : 2016).

II.4. Metode *Certainty Factor*

Certainty Factor (Theory) 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 informasi 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 (Muhammad Zunaidi, dkk ; 2015 : 188).

IF E1 [AND / OR] E2 [AND / OR] … En THEN H (CF = CFi) .....(1)

Keterangan :

E1 ... En : fakta – fakta (evidence) yang ada.

H : hipotesa atau konklusi yang dihasilkan.

II.5. Metode *Teorema Bayes*

Probabilitas Bayes merupakan salah satu cara untuk mengatasi ketidakpastian data dengan menggunakan formula bayes yang dinyatakan dengan rumus (Rika Rosnelly ; 2012 :79-80) :

$$P(H|E) = \frac{P(E|H).P(H)}{P(E)} \quad \textbf{.....(2)}$$

Dimana :

P(H | E) : *probabilitas hipotesis H* jika diberi *evidence E*

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

P(H) :*probabilitas hipotesis H* tanpa memandang *evidence* apapun

P(E) : *probabilitas evidence E*

Dalam bidang kedokteran *Teorema Bayes* sudah dikenal tetapi teorema lebih banyak diterapkan dalam logika kedokteran modern (Culter:1991). Teorema ini lebih banyak diterapkan pada hal-hal yang berkenaan dengan diagnosis secara

statistik yang berhubungan dengan probabilitas serta kemungkinan dari penyakit dan gejala-gejala yang berkaitan (Rika Rosnelly ; 2012 : 79-80).

II.6. *Microsoft Visual Studio* 2010

Visual Basic dibuat oleh *Microsoft*, merupakan salah satu bahasa pemrograman beriorentasi objek yang mudah dipelajari. Selain menawarkan kemudahan, *visual basic* juga cukup andal untuk digunakan dalam pembuatan berbagai aplikasi, terutama aplikasi *database*. *Visual basic* merupakan bahasa pemrograman *event drive*, dimana program akan menunggu sampai ada respons dari *user* / pemakai program aplikasi yang dapat berupa kejadian atau *event*, misalnya ketika *user* mengklik tombol atau menekan Enter. Jika kita membuat aplikasi dengan *visual basic* maka kita akan mendapatkan file yang menyusun aplikasi tersebut, yaitu :

1. *File Project (*.vbp)*

   File ini merupakan kumpulan dari aplikasi yang kita buat. *File project* biasa berupa *file *.frm, *.dsr* atau *file* lainnya.

2. *File Form (*.frm)*

   File ini merupakan *file* yang berfungsi untuk menyimpan informasi tentang bentuk *form* maupun *interface* yang kita buat (Edy Winarno,dkk ; 2013 : 83).

II.7. *Microsoft SQL Server* 2008

SQL Server 2008 adalah sebuah terobosan baru dar Microsoft dalam bidang *database*. *SQL Server* 2008 adalah sebuah 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 (Wahana Komputer ; 2013 : 2).

II.8. UML (*Unified Modelling Language)*

Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *Unified Modeling Language* (UML). UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, men-spesifikasikan dan membangun perangkat lunak.

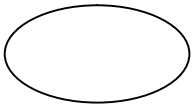
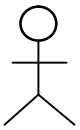
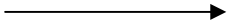
UML merupakan metodologi dalam mengembangkan sistem beriorentasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem. Alat bantu yang digunakan dalam perancangan beriorentasi objek berbasiskan UML.

II.8.1. *Use Case* Diagram

Use case diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat

dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fugsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* Diagram, yaitu :

Table II.1. *Use Case*

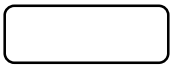
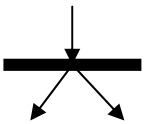
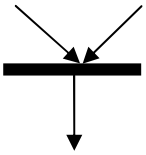
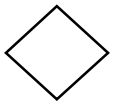
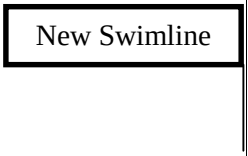
Gambar	Keterangan
	*Use Case* : peringkat tertinggi dari fungsionalitas yang dimiliki sistem.
	Aktor : Seseorang atau sesuatu yang berinteraksi dengan sistem yang sedang dikembangkan.
	Asosiasi antara aktor dan *use case*, di gambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasikan aliran data
	*Association* : adalah relasi antara a*ctor* dan *use case*
<<include>>	*Generalisasi* : untuk memperlihatkan struktur pewaris yang terjadi.
<<extend>>	*Extend* merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

(Sumber : Windu Gata ; 2013 : 5)

II.8.2. 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.2. *Activity Diagram*

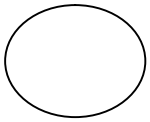
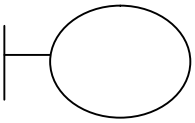
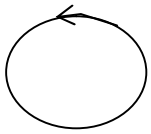
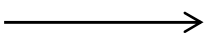
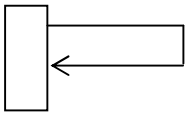
Simbol	Keterangan
	*Start Point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End Point*, akhir aktivitas.
	*Activities*, menggambarkan suatu proses/kegiatan bisnis.
	Faktor (percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau *Rake*, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, true atau false
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

(Sumber : Windu Gata ; 2013 : 6)

II.8.3. 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.3. *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. Kontrol objek mengkoordinir pesan antara *boundary* dengan entitas.
	*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)

II.8.4. diagram kelas *(Class Diagram*)

Merupakan hubungan antar kelas dan penjelasan detail tiap tiap kelas di dalam model desai, dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem.

Class diagram juga menunjukkan atribu-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class* diagram secara khas meliputi : kelas (*class*), relasi, *asssociation*, *generalization*, atribut (*attributes*), operasi (*operations/method*), dan *visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut.

Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau *kardinaliti*.

Tabel II.4. *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

(Sumber : Windu Gata ; 2013 : 8)

II.9. Normalisasi

Salah satu topik yang cukup kompleks dalam dunia manajemen *database* adalah proses untuk menormalisasi tabel-tabel dalam *database relasional.* Dengan normalisasi kita ingin mendesain *database relasional* yang terdiri dari tabel-tabel berikut :

1. Berisi data yang diperlukan.
2. Memiliki sesedikit mungkin redundansi.
3. Mengakomodasi banyak nilai untuk tipe data yang diperlukan.
4. Mengefisienkan *update*.
5. Menghindari kemungkinan kehilangan data secara tidak disengaja/tidak diketahui.

Alasan utama dari normalisasi *database* minimal sampai dengan bentuk normal ketiga adalah menghilangkan kemungkinan adanya "*insertion anomalies"*, "*deletion anomalies",* dan "*update anomalies"*. Tipe-tipe kesalahan tersebut sangat mungkin terjadi pada *database* yang tidak normal.

II.9.1. Bentuk - Bentuk Normalisasi

1. Bentuk Tidak Normal

Bentuk ini merupakan kumpulan data yang akan direkam, tidak ada keharusan mengikuti format tertentu, dapat saja tidak lengkap dan terduplikasi. Data dikumpulkan apa adanya sesuai keadaanya.

Tabel II.5. Normalisasi Tidak Normal

Kode	Kode Suplier	Nama Suplier	Kode Brg	Nama brg	Tgl
004	G01	Gobel Nustra	A01 A02	AC Model 1 AC Model 2	07/10/04
006	A03	Angkasa	B01 A03	Meja, Kursi	09/10/05

(Sumber : Kusrini ; 2012 : 44)

2. Bentuk Normal Tahap Pertama (1” Normal Form)

Definisi :

Sebuah tabel disebut 1NF jika :

a. Tidak ada baris yang duplikat dalam tabel tersebut.

b. Masing-masing cell bernilai tunggal

Catatan: Permintaan yang menyatakan tidak ada baris yang duplikat dalam sebuah tabel berarti tabel tersebut memiliki sebuah kunci, meskipun kunci tersebut dibuat dari kombinasi lebih dari satu kolom atau bahkan kunci tersebut merupakan kombinasi dari semua kolom.

Tabel II.6. Normalisasi Normal 1

Kode	Kode Suplier	Nama Suplier	Kode Brg	Nama brg	Tgl
004	G01	Gobel Nustra	A01	AC Model 1	07/10/04
004	G01	Gobel Nustra	A02	AC Model 2	07/10/04
006	A03	Angkasa	B01	Meja	09/10/05
006	A03	Angkasa	A03	Kursi	09/10/05

(Sumber : Kusrini ; 2012 : 45)

3. Bentuk Normal Tahap Kedua (2nd Normal Form)

Bentuk normal kedua (2NF) terpenuhi jika pada sebuah tabel semua atribut yang tidak termasuk dalam *primary key* memiliki ketergantungan fungsional pada primary key secara utuh.

KdFaktur	→	Tgl. Jtempo. KodeSup. NamaSup
KdFaktur,Kodebrg	→	namaBrg, Qty, Harga

4. Bentuk Normal Tahap Ketiga (3rd Normal Form)

Sebuah tabel dikatakan memenuhi bentuk normal ketiga (3NF), jika untuk setiap ketergantungan fungsional dengan notasi X -> A, dimana A mewakili semua atribut tunggal di dalam tabel yang tidak ada di dalam X, maka :

a. X haruslah superkey pada tabel tersebut.

b. Atau A merupakan bagian dari primary key pada tabel tersebut.

5. Bentuk Normal Tahap Keempat dan Kelima

Penerapan aturan normalisasi sampai bentuk normal ketiga sudah memadai untuk menghasilkan tabel berkualitas baik. Namun demikian, terdapat pula bentuk normal keempat (4NF) dan kelima (5NF). Bentuk Normal keempat berkaitan dengan sifat ketergantungan banyak nilai (*multivalued dependency*) pada suatu tabel yang merupakan pengembangan dari ketergantungan fungsional. Adapun bentuk normal tahap kelima merupakan nama lain dari *Project Join Normal Form* (PJNF).

6. Boyce Code Normal Form (BCNF)

a. Memenuhi 1st NF

b. Relasi harus bergantung fungsi pada atribut superkey (Kusrini ; 2012 ; 39)

KdFaktur	→	Tgl, Jtempo, KodeSup
KdSup	→	NamaSup
KdFaktur, Kodebrg	→	Qty, Harga
KodeBrg	→	NamaBrg