BAB II

LANDASAN TEORI

II.1. Sistem

Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan kegiatan atau untuk melakukan sasaran yang tertentu. Pendekatan sistem yang merupakan jaringan kerja dari prosedur lebih menekankan urutan-urutan operasi di dalam sistem. 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 sistem (*environment*) adalah 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 sistem merupakan media penghubung antara satu subsistem dengan subsistem lainnya. Melalui penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem lain.

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 (Jeperson Hutahaean ; 2014 : 2).

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 penyakit 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 Dasar 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) (Andri Saputra, 2011 : 203)

II.2.2. Basis Pengetahuan Sistem Pakar

Inferensi digunakan dalam sistem pakar untuk memperoleh informasi terbaru dari informasi yang sudah ada. Diataranya :

1. *Forward Chaining*

   Adalah strategi inferensi yang dimulai dengan sekumpulan fakta, fakta baru yang diperoleh dengan menggunakan *rule*, dimana alasan yang digunakan sesuai dengan fakta yang ada, dan melanjutkan proses ini sampai goal diraih atau sampai tidak ada *rule* selanjutnya yang mempunyai alasan yang sesuai dengan fakta yang ada maupun fakta yang diketahui
2. *Backwad Chaining*

   Adalah strategi inferensi yang diperoleh untuk membuktikan suatu hipotesis dengan dukungan informasi (Rizki Fitriani Maiza ; 2010 : 10).

II.2.3. Aktivitas Sistem Pakar

Bentuk-bentuk ini memungkinkan para ahli untuk dapat mengambil keputusan 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 (Andri Saputra, 2011 : 203)

II.2.4. Bentuk Sistem Pakar

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.

Ada 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 terkandung didalam 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.

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 (Andri Saputra, 2011 : 203).

II.2.5. Struktur Sistem Pakar

Sistem pakar disusun oleh dua bagian utama, yaitu :

1. Lingkungan pengembangan (*development environment*), digunakan untuk memasukkan pengetahuan pakar ke dalam lingkungan sistem pakar.
2. Lingkungan konsultasi (*consultation environment*), digunakan oleh pengguna yang bukan pakar guna memperoleh pengetahuan pakar (Andri Saputra, 2011 : 204).

II.2.6. Ciri - Ciri Sistem Pakar

Sistem pakar merupakan program-program praktis yang menggunakan strategi heuristik yang dikembangkan oleh manusia untuk menyelesaikan permasalahan-permasalahan yang spesifik (khusus), disebabkan oleh keheuristikannya dan sifatnya yang berdasarkan pada pengetahuan sehingga umumnya sistem pakar mempunyai ciri-ciri sebagai berikut :

1. Terbatas pada domain keahlian tertentu

2. Berdasarkan pada kaidah/*rule* tertentu.
3. Dapat digunakan dalam berbagai jenis komputer.
4. Mudah dimodifikasi, yaitu dengan menambah atau menghapus suatu kemampuan dari basis pengetahuannya.
5. Sistem dapat mengaktifkan kaidah secara searah yang sesuai, dituntun oleh dialog dengan pemakai (Sri Rahayu ; 2013 : 130)

II.2.7. Arsitektur Sistem Pakar

Arsitektur sistem pakar dapat dilihat pada gambar 1 di bawah ini dimana sebuah sistem pakar terdiri dari tiga modul utama, yaitu: knowledge base, working memory dan inference engine yang merupakan bagian utama dari sebuah sistem pakar. Sedangkan bagian-bagian selain ketiga komponen utama itu adalah : *user interface, developer interface, explanation facility,* dan *external programs*.

a. *Knowledge base* adalah representasi pengetahuan dari seorang atau beberapa pakar yang diperlukan untuk memahami, memformulasikan dan memecahkan masalah. Dalam hal ini digunakan untuk memecahkan masalah-masalah yang terjadi pada komputer. *Knowledge base* ini terdiri dari dua elemen dasar, yaitu fakta dan *rules*.
b. *Inference engine* merupakan otak dari sistem pakar yang mengandung mekanisme fungsi berpikir dan pola-pola penalaran sistem yang digunakan oleh seorang pakar. Mekanisme ini yang menganalisis suatu masalah tertentu dan kemudian mencari solusi atau kesimpulan yang terbaik.
c. *Working Memory* merupakan tempat penyimpanan fakta-fakta yang diketahui dari hasil menjawab pertanyaan.
d. *User/developer interface*. Semua *software* pengembangan sistem pakar memberikan interface yang berbeda bagi *user* dan *developer*. *User* akan berhadapan dengan

tampilan yang sederhana dan mudah sedangkan *developer* akan berhadapan dengan editor dan *source code* waktu mengembangkan program.

e. *Explanation facility* memberikan penjelasan saat mana *user* mengetahui apakah alasan yang diberikan sebuah solusi.

f. *External programs*. Berbagai program *seperti database, spreadsheets, algorithms*, dan lainnya yang berfungsi untuk mendukung sistem (Andreas Handojo ; 2013 : 4).

II.3. Metode *Theorema Bayes*

Probabilitas Bayes merupakan salah satu cara yang baik untuk mengatasi ketidakpastian data dengan menggunakan formula bayes yang dinyatakan dengan rumus :

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

Keterangan :

P(H | E) : probabilitas hipotesis H jika diberikan evidence E

P(E | H) : probabilitas munculnya evidence apapun

P(E) : probabilitas evidence E

Dalam bidang kedokteran teorema Bayes sudah dikenal tapi teorema ini lebih banyak diterapkan dalam logika kedokteran modern. Teorema ini lebih banyak diterapkan pada halhal yang berkenaan dengan probabilitas serta kemungkinan dari penyakit dan gejala-gejala yang berkaitan.

Misalnya seseorang menjalani tes klinik tersebut dan mendapatkan hasil positif, berapakah peluang bahwa ia benar-benar menderita penyakit langka

tersebut? Dengan kata lain, kita mencoba untuk mencari peluang dari A, dimana B atau P (A | B).

Dari tabel di atas, dapat kita lihat bahwa P (A | B) adalah peluang dari positif yang benar dibagi dengan peluang positif (benar maupun salah), yaitu 0,0194 / (0,0194 + 0,0882) = 0,1803. Kita dapat juga mendapatkan hasil yang sama dengan menggunakan rumus teorema Bayes di atas:

$$\boldsymbol{P}(A \mid B) = \frac{\boldsymbol{P}(B \cap A)}{\boldsymbol{P}(B)}$$

$$= \frac{\boldsymbol{P}(B \mid A) \times \boldsymbol{P}(A)}{\boldsymbol{P}(B \mid A)\boldsymbol{P}(A) + \boldsymbol{P}(B \mid A)\boldsymbol{P}(A)}$$

$$= \frac{97\% \times 2\%}{(97\% \times 2\%) + (9\% \times 98\%)}$$

$$= \frac{0.0194}{0.0194 + 0.0882}$$

$$= \frac{0.0194}{0.1076}$$

$$\boldsymbol{P}(A \mid B) = 0.1803$$

Hasil perhitungan ini sangat berbeda dengan intuisi kita di atas. Peluang bahwa orang yang mendapat hasil tes positif itu benar-benar menderita penyakit langka tidak sebesar yang kita bayangkan. Cuma ada sekitar 18% kemungkinan bahwa dia benar-benar menderita penyakit itu (Sri Rahayu ; 2013 : 131).

III.3.1. Langkah - Langkah Metode *Theorema Bayes*

Dalam melakukan perhitungan metode theorema bayes, maka diperlukan beberapa langkah – langkah dalam perhitungan metode theorema bayes seperti berikut:

1. Menentukan kriteria yang akan dinilai terlebih dahulu.
2. Kemudian menentukan data training.
3. Setelah data kriteria dan data training telah ditentukan, maka menentukan probabilitas dari setiap variabel.
4. Dari penelaian probabilitas maka akan dapat ditentukan nilai dari setiap karakter dan kriteria yang akan dinilai tersebut.
5. Kemudian dari nilai kriteria dapat diteruskan dengan mencari likelihood dan probabilitas dari masing – masing kriteria.
6. Adapun perhitungan mencari lokelihood dan probablitas dengan menggunakan rumus $P(H|E)=\frac{P(E|H).P(H)}{P(E)}$
7. Kemudian dapat ditentukan nilai tertinggi dari setiap data (Sri Rahayu ; 2013 : 131).

II.4. Database

Secara sederhana *database* (basis data/pangkalan data) dapat diungkapkan sebagai suatu pengorganisasian data dengan bantuan komputer yang memungkinkan data dapat diakses dengan mudah dan cepat. Pengertian akses dapat mencakup pemerolehan data maupun pemanipulasian data seperti menambah serta menghapus data. Dengan memanfaatkan komputer, data dapat

disimpan dalam media pengingat yang disebut *harddisk.* Dengan menggunakan media ini, keperluan kertas untuk menyimpan data dapat dikurangi. Selain itu, data menjadi lebih cepat untuk diakses terutama jika dikemas dalam bentuk *database* (Agustinus Mujilan ; 2012 : 23)

II.5. MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa inggris; *database management system*) data DBMS yang *multithread*, MySQL tersedia sebagai perangkat lunak gratis di bawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.

Tidak seperti Apache yang merupakan *software* yang dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing masing, MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial swedia yaitu MySQL AB. MySQL AB memegang penuh hak cipta hampir atas semua kode sumbernya (Achmad Solichin ; 2010 : 2).

II.6. Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai jenis komputer dan berbagai sistem operasi termasuk telepon genggam. Java dikembangkan oleh *Sun Microsystem* dan dirilis tahun 1995. Java merupakan suatu teknologi perangkat lunak yang digolongkan *multi platform*. Selain itu, Java

juga merupakan suatu *platform* yang memiliki *virtual machine* dan *library* yang diperlukan untuk menulis dan menjalankan suatu program.

Bahasa pemrograman java pertama lahir dari *The Green Project*, yang berjalan selama 18 bulan, dari awal tahun 1991 hingga musim panas 1992. Proyek tersebut belum menggunakan *versi* yang dinamakan Oak. Proyek ini dimotori oleh Patrick Naughton, Mike Sheridan, James Gosling dan Bill Joy, serta Sembilan pemrograman lainnya dari *Sun Microsystem.* Salah satu hasil proyek ini adalah mascot Duke yang dibuat oleh Joe Palrang (Wahana Komputer ; 2010 : 1)

II.7. Teknik 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.7.1. 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.

2. **Bentuk normal tahap pertama (1" Normal Form)**

   Definisi :

   Sebuah table 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.

3. **Bentuk normal tahap kedua (2^{nd} 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.

4. **Bentuk normal tahap ketiga (3^{rd} 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 ; 2010 : 39-43).

II.8. 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.

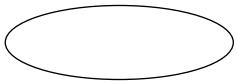
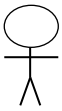
UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem.

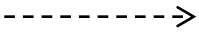
Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah *use case* diagram, *activity* diagram, *sequence* diagram dan *class* diagram.

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, yaitu :

Tabel II.2. 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.

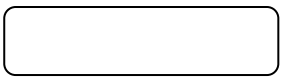
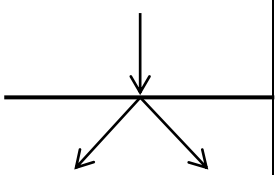
	Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *use case*, tetapi tidak memiliki control terhadap *use case*.
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara 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.

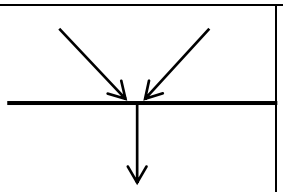
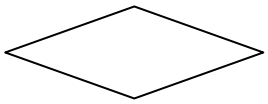
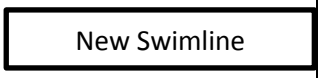
(Sumber : Windu Gata ; 2013 : 4)

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.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.

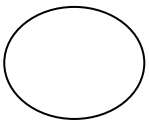
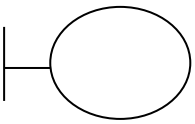
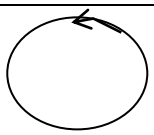
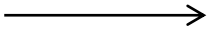
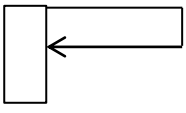
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true*, *false.*
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

(Sumber : 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.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 : 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.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 : Windu Gata ; 2013 : 9)