BAB II

TINJAUAN PUSTAKA

II.1. Sistem

Sistem merupakan sekumpulan elemen-elemen yang saling terintegrasi serta melaksanakan fungsinya masing-masing untuk mencapai tujuan yang telah di tetapkan. (Sulindawati; 2010 :1)

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 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 sistem adalah apapun diluar batas dari sistem yang mempengaruhi operasi sistem. Lingkungan 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 subsistem ke subsistem lainnya.

5. Masukan Sistem

Masukan sistem adalah energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan (*maintenance input*) dan masukan sinyal (*signal input*). *Maintenance input* adalah energi yang dimasukkan supaya sistem tersebut dapat beroperasi. Signal input adalah energi yang diproses untuk mendapatkan keluaran.

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. Sasaran dari sistem sangat menentukan sekali masukan yang dibutuhkan sistem dan keluaranya yang akan dihasilkan sistem.

(Sulindawati; 2010: 1-2)

II.2. Pakar

Pakar adalah seseorang yang mempunyai pengetahuan, pengalaman, dan metode khusus, serta mampu menerapkannya untuk memecahkan masalah atau memberi nasihat. Seorang pakar harus mampu menjelaskan dan mempelajari hal-hal baru yang berkaitan dengan topik permasalahan, jika perlu harus mampu menyusun kembali pengetahuan-pengetahuan yang didapatkan, dan dapat memecahkan aturan-aturan serta menentukan relevansi kepakarannya. Jadi seseorang pakar harus mampu melakukan kegiatan-kegiatan berikut :

1. Mengenali dan memformulasikan permasalahan.
2. Memecahkan permasalahan secara cepat dan tepat.
3. Menerangkan pemecahannya.
4. Belajar dari pengalaman.
5. Merestrukturisasi pengetahuan.
6. Memecahkan aturan-aturan.
7. Menentukan relevansi.(T. Sutojo; 2011: 163-164)

Selain itu pakar juga memiliki kemampuan untuk mengaplikasikan pengetahuannya dan memberikan saran serta pemecahan masalah pada domain tertentu. Seorang pakar mengetahui fakta-fakta mana yang penting, sebab akibat, fenomena-fenomena yang terkait dengan fakta, memahami arti hubungan antar fakta, juga hubungan sebab akibat, dan hubungan dengan fenomena-fenomena yang terkait serta mampu menginterprestasikan akibat-akibat yang terjadi karena sesuatu sebab terjadi. (Rika Rosnelly; 2012: 10)

II.3. Sistem Pakar

II.3.1. Pengertian Sistem Pakar

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.

Sebuah sistem pakar memiliki 2 komponen utama yaitu basis pengetahuan (*knowledge based*), berisi pengetahuan yang akan digunakan oleh komponen lainnya yaitu mesin referensi (*inference engine*) untuk menghasilkan kesimpulan sebagai respon terhadap *query* yang dilakukan *user*.

Sebuah sistem pakar umumnya tidak memiliki pengetahuan lain diluar area pengetahuannya kecuali jika diprogram dan dimuat kedalam sistem. Misalkan sebuah sistem pakar yang memuat pengetahuan mengenai penyakit infeksi mungkin tidak memiliki pengetahuan lain dalam area masalah kedokteran. Dalam area pengetahuan yang dimiliki, sistem pakar melakukan referensi atau membuat kesimpulan dengan cara yang sama seperti seorang pakar menarik kesimpulan. (Rika Rosnelly ; 2012 ; 2-4).

II.3.2. Manfaat Sistem Pakar

Sistem pakar menjadi sangat populer karena sangat banyak kemampuan dan manfaat yang diberikannya, diantaranya:

1. Meningkatkan produktivitas, karena sistem pakar dapat bekerja lebih cepat daripada manusia.

2. Membuat seorang yang awam bekerja seperti layaknya seorang pakar.
3. Meningkatkan kualitas, dengan memberi nasehat yang konsisten dan mengurangi kesalahan.
4. Mampu menangkap pengetahuan dan kepakaran seseorang.
5. Dapat beroperasi di lingkungan yang berbahaya.
6. Memudahkan akses pengetahuan seorang pakar.
7. Andal.
8. Meningkatkan kapabilitas sistem komputer.
9. Mampu bekerja dengan informasi yang tidak lengkap atau tidak pasti.
10. Bisa digunakan sebagai media pelengkap dalam penelitian.
11. Meningkatkan kemampuan untuk menyelesaikan masalah karena sistem pakar mengambil sumber pengetahuan dari banyak pakar.

    (T. Sutojo; 2011: 160)

II.3.3. Kekurangan Sistem Pakar

Selain manfaat, ada juga beberapa kekurangan yang ada pada sistem pakar, diantaranya :

1. Biaya yang sangat mahal untuk membuat dan memeliharanya.
2. Sulit dikembangkan karena keterbatasan keahlian dan ketersediaan pakar.
3. Sistem pakar tidak 100% bernilai benar. (T. Sutojo; 2011: 161)

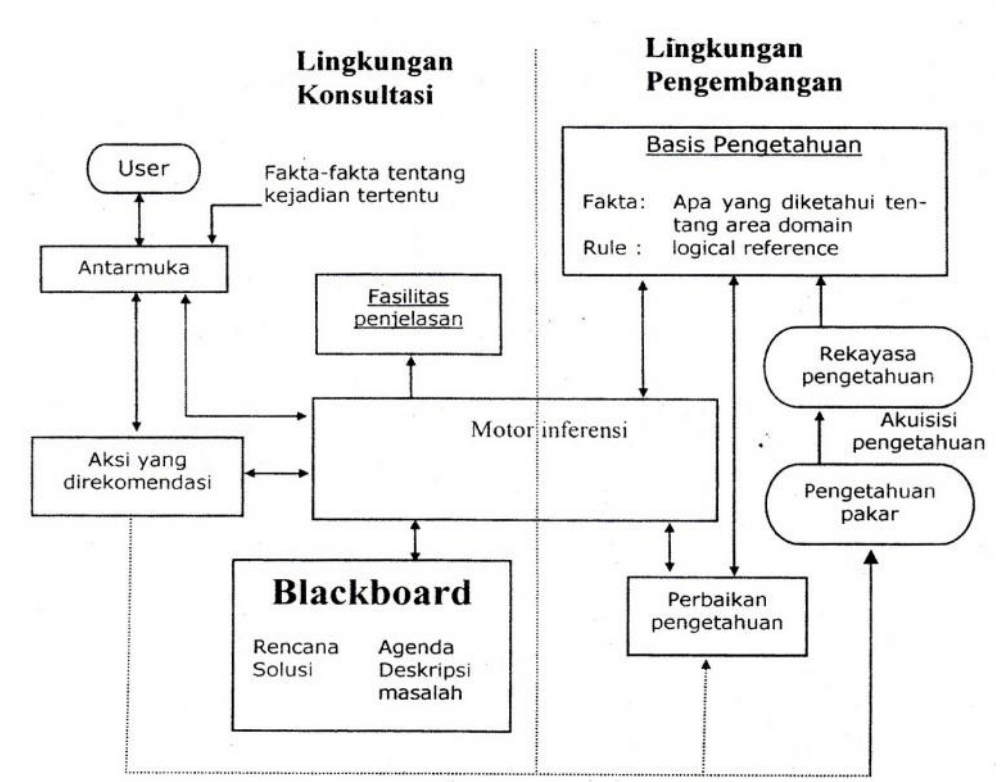
II.3.4. Ciri-ciri Sistem Pakar

Ciri-ciri dari sistem pakar adalah sebagai berikut:

1. Terbatas pada domain 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 searah yang sesuai, dituntun oleh dialog dengan pengguna. (T. Sutojo; 2011: 162)

II.3.5. Struktur Sistem Pakar

Ada dua bagian penting dari sistem pakar, yaitu lingkungan pengembangan (*development environment*) dan lingkungan konsultasi (*consultation environment*). Lingkungan pengembangan digunakan oleh pembuat sistem pakar untuk membangun komponen-komponennya dan memperkenalkan pengetahuan ke dalam *knowledge base* (basis pengetahuan). Lingkungan konsultasi digunakan oleh pengguna untuk berkonsultasi sehingga pengguna mendapatkan pengetahuan dan nasihat dari Sistem Pakar layaknya berkonsultasi dengan seorang pakar. Gambar II.1 menunjukkan komponen-komponen yang penting dalam sebuah sistem pakar:



Gambar II.1 : Struktur Sistem Pakar

Sumber: (T. Sutojo; 2011: 167)

Keterangan:

1. Akuisisi Pengetahuan

   Subsistem ini digunakan untuk memasukkan pengetahuan dari seorang pakar dengan cara merekayasa pengetahuan agar bisa diproses oleh komputer dan manaruhnya ke dalam basis pengetahuan dengan format tertentu (dalam bentuk representasi pengetahuan). Sumber-sumber pengetahuan bisa diperoleh dari pakar, buku, dokumen, multimedia, basis data, laporan riset khusus, dan informasi yang terdapat di web.

2. Basis Pengetahuan (*Knowledge Base*)

Basis pengetahuan mengandung pengetahuan yang diperlukan untuk memahami, memformulasikan, dan menyelesaikan masalah. Basis pengetahuan terdiri dari dua elemen dasar, yaitu:

a. Fakta, misalnya situasi, kondisi, atau permasalahan yang ada.
b. Rule (aturan), untuk mengarahkan penggunaan pengetahuan dalam memecahkan masalah.

3. Mesin Inferensi (*Inference Engine*)

Mesin inferensi adalah sebuah program yang berfungsi untuk memandu proses penalaran terhadap suatu kondisi berdasarkan pada basis pengetahuan yang ada, memanipulasi dan mengarahkan kaidah, model, dan fakta yang disimpan dalam basis pengetahuan untuk mencapai solusi atau kesimpulan. Dalam prosesnya, mesin inferensi menggunakan strategi pengendalian, yaitu strategi yang berfungsi sebagai panduan arah dalam melakukan proses penalaran. Ada tiga teknik pengendalian yang digunakan, yaitu *forward chaining, backward chaining,* dan gabungan dari kedua teknik tersebut.

4. Daerah Kerja (*Blackboard*)

Untuk merekam hasil sementara yang akan dijadikan sebagai keputusan dan untuk menjelaskan sebuah masalah yang sedang terjadi, sistem pakar membutuhkan *Blackboard* yaitu area pada memori yang berfungsi sebagai basis data. Tiga tipe keputusan yang dapat direkam pada *Blackboard*, yaitu:

a. Rencana : bagaimana menghadapi masalah.
b. Agenda : aksi-aksi potensial yang sedang menunggu untuk dieksekusi.

c. Solusi : calon aksi yang akan dibangkitkan.

5. Antarmuka Pengguna (*User Interface*)

Digunakan sebagai media komunikasi antara pengguna dan sistem pakar, komunikasi ini paling bagus bila disajikan dalam bahasa alami (*natural language*) dan dilengkapi dengan grafik, menu, dan formulir elektronik. Pada bagian ini akan terjadi dialog antara sistem pakar dan pengguna.

6. Subsistem Penjelasan (*Explanation Subsystem/Justifire*)

Berfungsi memberi penjelasan kepada pengguna, bagaimana suatu kesimpulan dapat diambil. Kemampuan seperti ini sangat penting bagi pengguna untuk mengetahui proses pemindahan keahlian pakar maupun dalam pemecahan masalah.

7. Sistem Perbaikan Pengetahuan (*Knowledge Refining System*)

Kemampuan memperbaiki pengetahuan (*knowledge refining system*) dari seorang pakar diperlukan untuk menganalisis pengetahuan, belajar dari kesalahan masa lalu, kemudian memperbaiki pengetahuannya sehingga dapat dipakai pada masa mendatang. Kemampuan evaluasi diri seperti itu diperlukan oleh program agar dapat menganalisa alasan-alasan kesuksesan dan kegagalannya dalam mengambil kesimpulan. Dengan cara ini basis pengetahuan yang lebih baik dan penalaran yang lebih efektif akan dihasilkan.

8. Pengguna (*User)*

Pada umumnya pengguna sistem pakar bukanlah seorang pakar (*non-expert*) yang membutuhkan solusi, saran, atau pelatihan (*training*) daru berbagai permasalahan yang ada. (T. Sutojo; 2011: 169).

II.4. *Penyakit Pada Unggas*

1. *Avian Influenza* (AI)

AI merupakan penyakit yang disebabkan oleh virus. Tanda-tanda umum itik yang terserang penyakit AI adalah keluar air mata, bersin-bersin, keluar cairan dari hidung, dan pembengkakan mucus dan menyebabkan kotoran pada mata. Penyakit ini dapat menular dari satu ternak ke ternak lain dan tidak ada pengobatannya. Vaksin AI sudah tersedia di pasaran namun efektivitas dari vaksin tersebut masih terus dilakukan penelitian.

2. *Duck Cholera*

Penyakit ini biasanya disebabkan oleh bakteri *Pasteurellamultocida* dan sangat merugikan peternak karena dapat menyebabkan kematian yang tinggi pada itik darat dan petelur. Anak itik berumur 4 minggu ke atas sangat peka terhadap penyakit ini. Anak itik yang terserang penyakit ini memiliki gejala-gejala diare dan biasanya disertai dengan sesak napas. Itik yang terjangkit penyakit *duck cholera* harus segera dipisahkan. Penyakit *Duck Cholera* dapat diobati dengan preparat sulfa dan antibiotik. Selain itu, penyakit ini bisa dicegah dengan sanitasi dan manajemen yang baik.

3. *Salmonellosis*

Penyakit *Salmonellosis* disebabkan oleh beberapa tipe *salmonella*. Itik yang terserang penyakit ini menunjukkan tanda-tanda tidak bergairah, dehidrasi yang disertai diare, kehilangan keseimbangan, kepala gemetar dan terputar. Ciri spesifik penyakit ini yaitu tampak pada *caecum* (usus) yang membengkak dan ada tonjolan, *mucosa* (selaput) di sekitar *rectum* membengkak dan terdapat cairan keputih-putihan, itik yang sudah sembuh biasanya pertumbuhannnya terhambat dan terdapat luka-luka pada ususnya. Walaupun itik sudah sembuh, tetapi masih dapat mengeluarkan bibit penyakit ke itik lain melalui kotorannya, sehingga harus dipisahkan dari kelompok. Penyakit yang disebabkan faktor makanan dan lingkungan penyakit pada kelompok ini antara lain keracunan, kekurangan zat-zat makanan, stres dan *afl atoksicosis* yang disebabkan oleh racun *afl atoksin* yang terdapat dalam bahan pakan seperti bungkil kacang tanah dan jagung.

(Sukmaya ; 2010 ; 18-19).

II.5. *Dempster-Shafer*

Metode *Dempster-Shafer* pertama kali diperkenalkan oleh Dempster, yang melakukan percobaan model ketidakpastian dengan *range* probabilities dari pada sebagai probabilitas tunggal. Kemudian pada tahun 1976 Shafer mempublikasikan teori Dempster itu pada sebuah buku yang berjudul *Mathematical Theory Of Evident*. *Dempster-Shafer Theory Of Evidence*, menunjukkan suatu cara untuk memberikan bobot kenyakinan

sesuai fakta yang dikumpulkan. Pada teori ini dapat membedakan ketidakpastian dan ketidaktahuan. Teori *Dempster-Shafer* adalah representasi, kombinasi dan propogasi ketidakpastian, dimana teori ini memiliki beberapa karakteristik yang secara instutitif sesuai dengan cara berfikir seorang pakar, namun dasar matematika yang kuat.

Ada berbagai macam penalaran dengan model yang lengkap dan sangat konsisten, tetapi pada kenyataannya banyak permasalahan yang tidak dapat terselesaikan secara lengkap dan konsisten. Ketidak konsistenan yang tersebut adalah akibat adanya penambahan fakta baru. Penalaran yang seperti itu disebut dengan penalaran *non monotonis*. Untuk mengatasi ketidak konsistenan tersebut maka dapat menggunakan penalaran dengan teori *Dempster-Shafer*. Secara umum teori Dempster-Shafer ditulis dalam suatu interval.

Penulisan umum: [*belief, plausibility*]

1. *Belief* (Bel) adalah ukuran kekuatan *evidence* dalam mendukung suatu himpunan proposisi. Jika bernilai 0 maka mengindikasikan bahwa tidak ada *evidence*, dan jika bernilai 1 menunjukkan adanya kepastian.
2. *Plausibility* (P1) dinotasikan sebagai:

   Pl(s) = 1 – Bel(⌐s)

   Plausibility juga bernilai 0 sampai 1. Jika yakin akan ⌐s, maka dapat dikatakan bahwa Bel (⌐s) = 1, dan PI (⌐ s) = 0.

Pada teori *Dempster-Shafer* dikenal adanya *frame of discernment* yang dinotasikan dengan θ. Frame ini merupakan semesta pembicaraan dari sekumpulan hipotesis. Tujuannya adalah mengkaitkan ukuran

kepercayaan elemen-elemen θ. Tidak semua *evidence* secara langsung mendukung tiap-tiap elemen. Untuk itu perlu adanya probabilitas fungsi densitas (m). Nilai m tidak hanya mendefinisikan elemen-elemen θ saja, namun juga semua subsetnya. Sehingga jika θ berisi nelemen, maka subset θ adalah 2n. Jumlah semua m dalam subset θ sama dengan 1. Apabila tidak ada informasi apapun untuk memilih hipotesis, maka nilai: m{θ} = 1,0.

Apabila diketahui X adalah subset dari θ, dengan m1 sebagai fungsi densitasnya, dan Y juga merupakan subset dari θ dengan m2 sebagai fungsi densitasnya, maka dapat dibentuk fungsi kombinasi m1 dan m2 sebagai m3, yaitu:

$$M_3(Z) = \frac{\sum_{X \cap Y = Z} m_1(X).m_2(Y)}{1 - K} \quad \text{..... (1)}$$

Keterangan:

m1 (X) = *mass function dari evidence* X

m2 (Y) = *mass function dari evidence* Y

m3 (Z) = *mass function dari evidence* Z

κ = jumlah *conflict evidence* (Mustikadewi P; 2012: 3)

II.6. Pengertian 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.

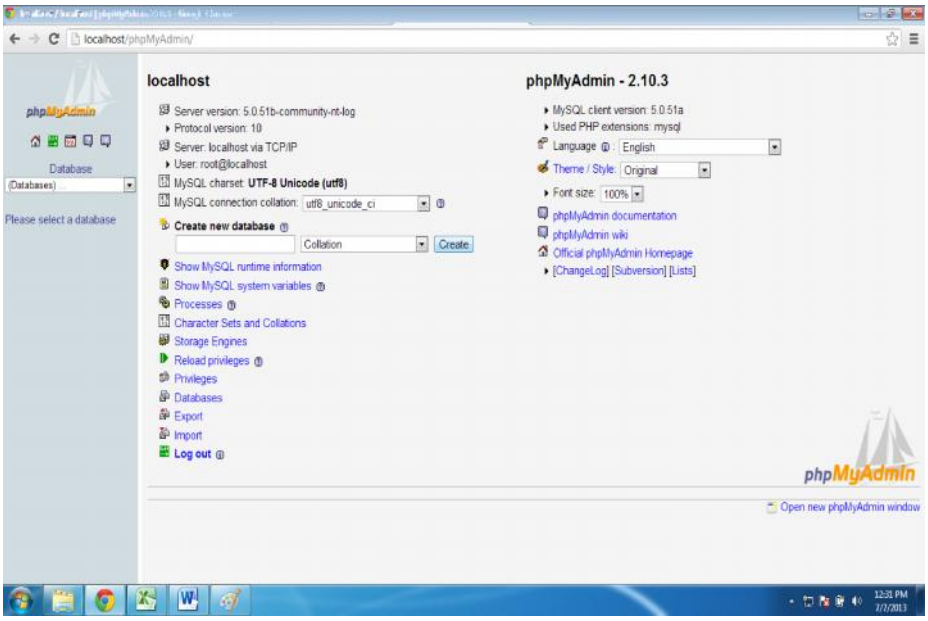
Bahasa pemrograman java pertama lahir dari *The Green Project*, yang berjalan selama 18 bulan, dari awal tahun 1991 hingga musim panas 1992. Proyek tersebut belum menggunakan *versi* yang dinamakan *Oak*. Proyek ini dimotori oleh Patrick Naughton, Mike Sheridan, James Gosling dan Bill Joy, serta Sembilan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

II.7. 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).



Gambar II.2. Tampilan MySQL
(Sumber : Wahana Komputer, 2010 : 5)

II.8. UML *(Unified Modelling 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.

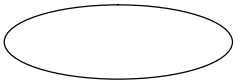
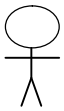
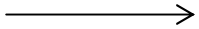
UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem.

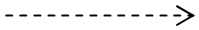
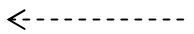
Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

1. *Use case* Diagram

Use case diagram merupakan pemodelan untuk 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.1. Simbol *Use Case*

Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	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.

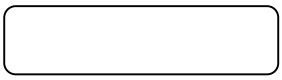
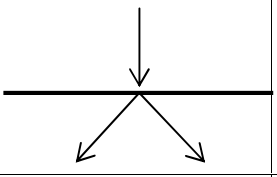
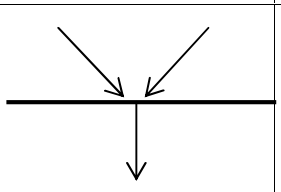
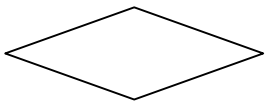
	*Include*, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	*Extend*, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

(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.2. Simbol *Activity Diagram*

Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true*, *false*.

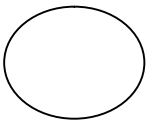
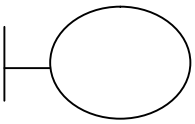
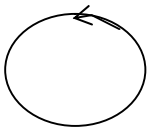
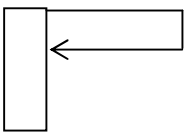
New Swimline	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

(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.3. Simbol *Sequence Diagram*

Gambar	Keterangan
	*Entity Class*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*Boundary Class*, berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan 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.4. *Multiplicity Class Diagram*

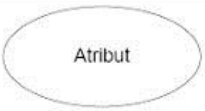
Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : Windu Gata, 2013 : 9)

II.8. *Entity Relationship Diagram* (ERD)

Entity Relationship Diagram atau ERD adalah alat pemodelan data utama dan akan membantu mengorganisasi data dalam suatu proyek ke dalam entitas-entitas dan menentukan hubungan antarentitas. Proses memungkinkan analis menghasilkan struktur basisdata yang baik sehingga data dapat disimpan dan diambil secara efisien (Janner Simarmata, 2010 : 67).

Tabel II.5. Simbol ERD

Notasi	Keterangan
	**Entitas**, adalah suatu objek yang dapat diidentifikasi dalam lingkungan pemakai.
	**Relasi**, menunjukkan adanya hubungan di antara sejumlah entitas yang berbeda.
	**Atribut**, berfungsi mendeskripsikan karakter entitas (atribut yg berfungsi sebagai key diberi garis bawah)
	Garis, sebagai penghubung antara relasi dengan entitas, relasi dan entitas dengan atribut.

(Sumber : Janner Simarmata, 2010 : 67)

II.9. 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.

II.9.1.Bentuk-bentuk Normalisasi

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.

Supaya bisa menggabungkan jumlah barang yang di pasok (qty) secara unik dengan barang (b#) dan pemasok (#p), kita menggunakan kunci utama gabungan yang tersusun dari b# dan p#. Sebuah tabel relasional secara definisi selalu berada dalam bentuk normal pertama. Semua nilai pada kolom-kolom nya adalah atomik. Ini berarti kolom-kolom tidak mempunyai nilai berulang . Gambar 2.3 menunjukkan tabel pemasok dalam 1NF.(Janner Simarmata; 2010 : 79).

P#	Status	kota	B#	Qty
P1	20	Yogkarta	B1	300
P1	20	Yogkarta	B2	200
P1	20	Yogkarta	B3	400
P1	20	Yogkarta	B4	200
P1	20	Yogkarta	B5	100
P1	20	Yogkarta	B6	100
P2	10	Medan	B1	300
P2	10	Medan	B2	400
P3	10	Medan	B2	200
P4	20	Yogkarta	B2	200
P4	20	Yogkarta	B2	300
P4	20	Yogkarta	B5	400

Gambar 2.3 Tabel bentuk normal pertama (1NF)

(Sumber : Janner Simarmata, 2010 : 80)

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.

Tabel pemasok berada pada INF, tetapi tidak pada 2NF karena status dan kota tergantung secara fungsional hanya pada kolom p# dari kunci gabungan (p#, b#). Ini dapat digambarkan dengan membuat daftar ketergantungan fungsional;

p# : kota, status

kota : status

(p#, b#) : qty

Pada contoh, kita memindahkan kolom p#, status,dan kota ke tabel baru yang disebut PEMASOK2. kolom p# menjadi kunci utama tabel ini. Gambar 2.4 menunjukkan hasilnya.(Janner Simarmata; 2010 ; 81-82).

Pemasok2

P#	Status	Kota
P1	20	Yogkarta
P2	10	Medan
P3	10	Medan
P4	20	Yogkarta
P5	30	Bandung

Barang

P#	B#	Qty
P1	B1	300
P1	B2	200
P1	B3	400
P1	B4	200
P1	B5	100
P2	B1	300
P2	B2	400
P3	B2	200
P4	B2	200
P4	B4	300
P4	B5	400

Gambar 2.4 Tabel bentuk normal kedua (2NF)

(Sumber : Janner Simarmata, 2010 : 82)

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.

Untuk mengubah PEMASOK2 menjadi 3NF, kita membuat tabel baru yang disebut KOTA_STATUS dan memindahkan kolom kota dan status ke tabel baru. Status dihapus dari tabel asal, kota tetap dibiarkan karena akan berfungsi sebagai kunci asing bagi KOTA_STATUS, dan tabel asal diberi nama baru

PEMASOK_KOTA. Gambar 2.5 menunjukkan hasilnya. (Andi; 2010 ; 82-83).

PEMASOK_KOTA

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

KOTA_STATUS

Kota	Status
Yogyakarta	20
Medan	10
Bandung	30
Semarang	40

Gambar 2.5 Tabel bentuk normal ketiga (3NF)

(Sumber : Janner Simarmata, 2010 : 82)

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.(Janner Simarmata; 2010 ; 84-85).

5. Bentuk Normal Tahap Keempat dan Kelima

Sebuah tabel relasional 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).

Misalnya, ada sebuah tabel relasional R dengan kolam A, B dan C, maka R.A$\rightarrow$ R.B (kolom A menentukkan kolom B). Adalah

benar jika dan hanya jika himpunan nilai B yang cocok dengan pasangan nilai A dan nilai C pada R hanya tergantung pada nilai A dan tidak tergantung pada nilai C.

Misalnya, pegawai ditugaskan kebanyak proyek dan ia mempunyai banyak keahlian. Jika kita mencatat informasi ini pada satu tabel, ketiga atribut harus digunakan sebagai kunci karena tidak ada satu atribut pun yang dapat secara unik mengidentifikasi sebauh *record*. untuk mengubah sebuah tabel dengan ketergantungan *multivalue* ke dalam 4NF, pindahkan masing-masing pasangan MVD ke tabel baru. Gambar 2.6 menunjukkan hasilnya.

PEGAWAI_PROYEK

Peg#	Pry#
1211	P1
1211	P3

PEGAWAI_AHLI

Peg#	Ahli
1211	Analisis
1211	Perancangan
1211	Pemrograman

Gambar 2.6 Tabel bentuk normal keempat (4NF)

(Sumber : Janner Simarmata, 2010 : 86)

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*).

contoh, misalkan kita mempunyai pegawai yang menggunakan keahlian perancangan pada suatu proyek lainnya. (Janner Simarmata, 2010 : 85-86).