BAB II

LANDASAN TEORI

II.1. Kecerdasan Buatan

Kecerdasan buatan berasal dari bahasa inggris "*Artificial Intelligence"* atau disebut AI, yaitu *Intelligence* adalah kata sifat yang berarti cerdas, sedangkan *Artificial* yang berarti buatan. Kecerdasan buatan yang dimaksud di sini merujuk pada mesin yang mampu berfikir, menimbang tindakan yang akan di ambil, dan mampu mengambil keputusan seperti yang dilakukan oleh manusia. (T. Sutojo; 2011: 1-2).

II.2. Sistem Pakar

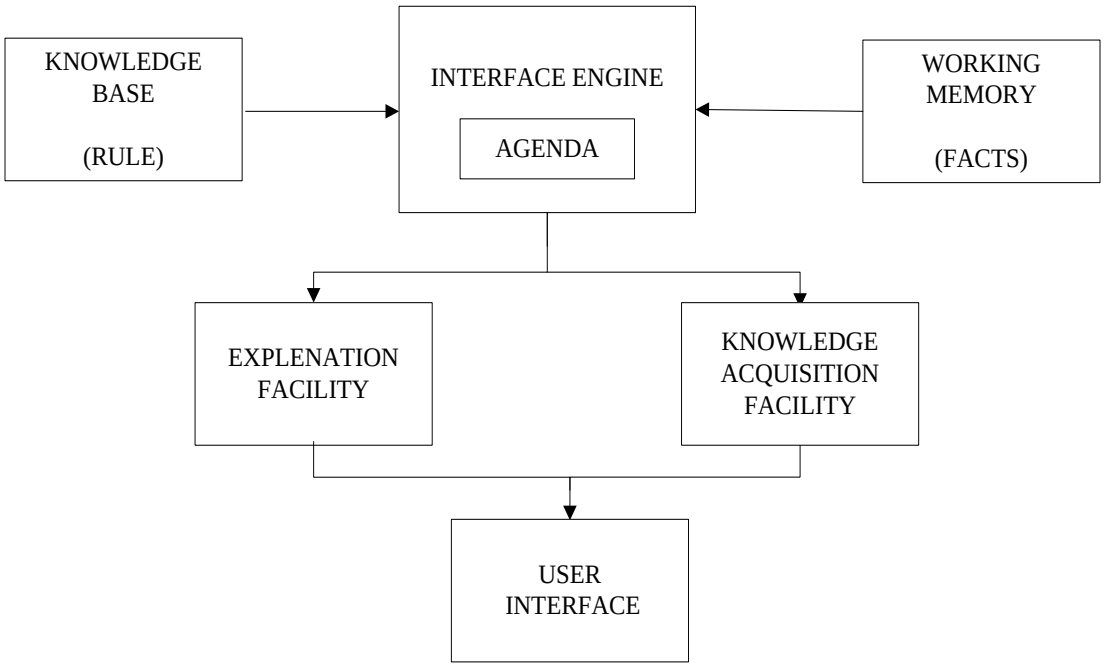
Bidang sistem pakar merupakan penyelesaian pendekatan yang sangat berhasil dan bagus untuk permasalahan AI klasik dari pemprograman *intelligent* (cerdas). Sistem Pakar *(expert system)* merupakan solusi AI *(Artificial Intelligence)* bagi permasalahan pemrograman pintar. Profesor Edward Feigenbaum dari Stanford University yang merupakan pionir dalam teknologi sistem pakar mendefinisikan 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. (Rika Rosnelly; 2012: 2).

Secara umum, sistem pakar (Expert System) adalah sistem yang berusaha mengadopsi pengetahuan manusia ke komputer, agar komputer dapat menyelesaikan masalah seperti yang biasa dilakukan oleh para ahli. Dengan sistem pakar ini, orang awam pun dapat menyelesaikan masalah yang cukup rumit yang sebenarnya hanya dapat diselesaikan dengan bantuan para ahli. Tujuan pengembangan sistem pakar sebenarnya tidak untuk menggantikan peran para pakar, namun untuk mengimplementasikan pengetahuan para pakar ke dalam bentuk perangkat lunak, sehingga dapat digunakan oleh banyak orang dan tanpa biaya yang besar. Untuk membangun sistem yang difungsikan untuk menirukan seorang pakar manusia harus bisa melakukan hal-hal yang dapat dikerjakan oleh para pakar. Untuk pembangun sistem yang seperti itu maka komponen-komponen dasar yang minimal harus dimiliki adalah sebagai berikut:

1. Antar muka (*user interface*).
2. Basis pengetahuan (*knowledge base*).
3. Mesin inferensi (*Inference Engine*). (Yasidar Nur Istiqomah; 2013 : 34).

II.2.1. Struktur Sistem Pakar

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



Gambar II.1. : Struktur Sistem Pakar
Sumber : (Rika Rosnelly; 2012: 12)

Komponen yang terdapat dalam struktur sistem pakar ini adalah *knowledge base (rules), inference engine, working memory, explanation facility, knowledge acquisition facility, dan user interface*

1. *Knowledge Base* (Basis pengetahuan)

   Basis pengetahuan mengandung pengetahuan untuk pemahaman, formulasi, dan penyelesaian masalah. Komponen sistem pakar disusun atas atas dua elemen dasar, yaitu fakta dan aturan. Fakta merupakan informasi tentang objek dalam area permasalahan tertentu, sedangkan aturan merupakan informasi tentang cara bagaimana memperoleh fakta baru dari fakta yang telah diketahui. Pada struktur sistem pakar diatas, knowledge basee disini untuk menyimpan pengetahuan dari pakar berupa rule / aturan (if <kondisi> then <aksi> atau dapat juga disebut condition-action rules).

2. *Inference Engine* (Mesin Inferensi)

   Mesin inferensi merupakan otak dari srbuah sistem pakar dan dikenal juga dengan sebuatan *control structure* (struktur control) *atau rule interpreter* (dalam sistem pakar berbasis kaidah). Komponen ini mengadung mekanisme pola pikir dan penalaran yang digunakan oleh pakar dalam menyelesaikan suatu masalah. Mesin inferensi disini adalah *processor* pada sistem pakar yang mencocokkan bagian kondisi dari rule yang tersimpan di dalam *knowledge base* dengan fakta yang tersimpan di *working memory*.

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 computer, yang bertujuan untuk memperbaiki atau mengembanngkan basis pengetahuan.

6. *User Interface*

   Mekanisme untuk memberi kesempatan kepada user dan sistem pakar untuk berkomunikasi. Antar muka menerima informasi dari pemakai dan mengubahnya kedalam bentuk yang dapat dapat diterima oleh sistem. Selain itu antar muka menerima informasi dari sistem dan menyajikannya kedalam bentuk yang dapat di mengerti oleh pemakai. (Rika Rosnelly; 2012: 10).

II.2.2. Pakar

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 :

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. (Rika Rosnelly; 2012: 10).

II.2.3. Manfaat Sistem Pakar

Sistem pakar menjadi sangat populer karena sangat banyak manfaat dari sistem pakar diantaranya :

1. Meningkatkan produktifitas, karena sistem pakar dapat bekerja lebih cepat dari pada manusia.
2. Membuat seseorang 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. Sistem pakar tidak pernah menjadi bosan dan kelelahan atau sakit.
8. Meningkatkan kapabilitas sistem komputer. Integrasi sistem pakar dengan sistem komputer lain membuat sistem lebih efektif dan mencakup lebih banyak aplikasi.
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; 2010: 160).

II.2.4. Kekurangan Sistem Pakar

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

1. Biaya yang sangat mahal untuk membuat dan memeliharanya.
2. Sulit dikembangkan karena keterbatasan keahlian dan memeliharanya.
3. Sistem pakar tidak 100% bernilai benar.

II.3. Lazy Eye (Mata malas)

Mata Malas (*lazy eye*) adalah kurang jelasnya penglihatan akibat perkembangan penglihatan yang tidak sempurna dalam otak. Otak manusia membutuhkan stimulasi visual untuk berkembang sepenuhnya. Pada saat perkembangan anak sejak lahir hingga usia 8 tahun, apapun yang menghalangi atau menggangu jelasnya penglihatan dapat menyebabkan *lazy eye*. Penyebab umum termasuk ukuran kacamata tinggi (contohnya astigmatisma, hiperopia dan miopia), mata juling (strabismus), atau apapun yang menghalangi aksis visual pada satu mata (contohnya kelopak mata turun, katarak anak). Mata malas (*lazy eye*) biasanya hanya mempengaruhi satu mata, tetapi apabila kedua mata kurang mendapat visual yang baik dan jelas untuk periode yang berkepanjangan, kondisi dapat timbul pada kedua mata. Diagnosa dini meningkatkan kemungkinan suksesnya pengobatan, karena setelah usia 8 tahun, kerusakan visual dapat menjadi permanen. Sebaliknya, jika anak anda tidak mengalami *lazy eye* hingga usia 8 tahun, maka kemungkinan untuk mengidap *lazy eye* sangatlah kecil. (http://www.snec.com.sg)

Adapun gejala – gejala *lazy eye* sebagai berikut:

1. Gangguan penglihatan karena cacat sejak lahir
2. Ptosis (mata turun)
3. Anak harus lebih maju saat menonton TV ataupun belajar di kelas
4. Sakit kepala yang sering dan berkepanjangan
5. Strabismus (mata juling)

6. Perbedaan pada ukuran kacamata
7. Hiperopia (rabun dekat)
8. Riwayat keluarga
9. Crowding phenomenon (penurunan tajam penglihatan)
10. Miopia (rabun jauh)
11. luka pada mata
12. Anisometropik (ukuran mata tidak setara)
13. Deprivasi
14. Isometropia
15. Adanya efek destinasi filter
16. Astigmatisma yang besar

II.4. Dempster Shafer

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: [*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.dimana nilai Bel yaitu (0-0.9).
2. *Plausibility* (Pl) 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 Pl (¬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 n elemen, 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: (Muhammad Dahria ; 2013)

$$m_3(Z) = \frac{\sum_{X \cap Y = Z} m_1(X).m_2(Y)}{1 - \sum_{X \cap Y = \phi} m_1(X).m_2(Y)} \quad \text{.....(1)}$$

Keterangan :

m = Nilai densitas (kepercayaan)

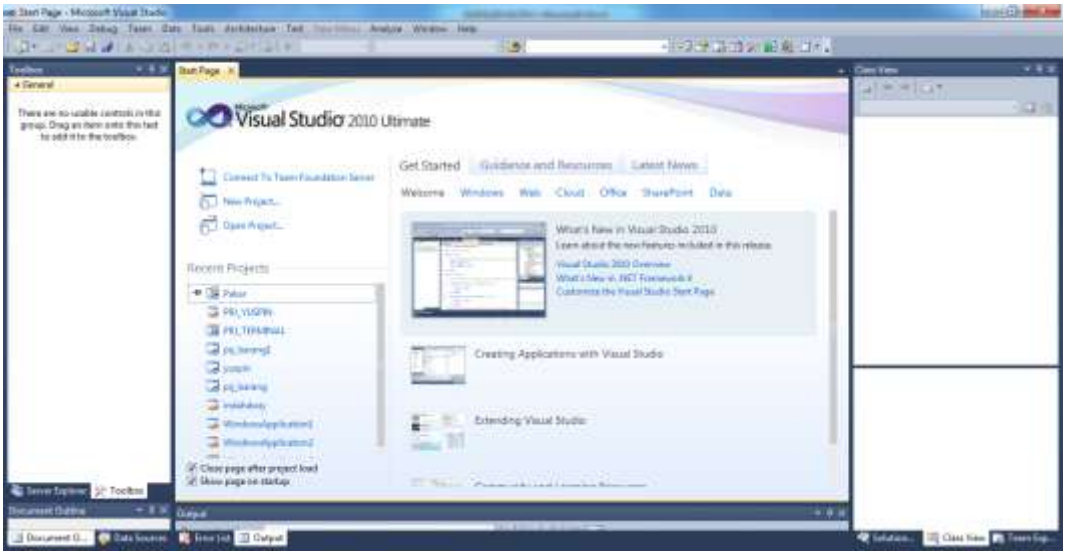
XYZ = Himpunan evidence

$\sum_{X \cap Y = Z} m_1(X).m_2(Y)$ = merupakan nilai kekuatan dari *evidence* Z yang diperoleh nilai keyakinan sekumpulan *evidence*.

II.5. Pemrograman Visual Basic 2010

Visual Basic 2010 merupakan suatu paket teknologi bahasa pemrograman yang dikeluarkan oleh microsoft. Bahasa pemrograman *Visual Basic* digunakan untuk membuat aplikasi windows yang berbasis *Graphical User Interface* (GUI). Microsoft *Visual Basic* 2010 sebagai produk IDE (*Integrated Development Environments)* andalan yang dikeluarkan Microsoft. Microsoft *Visual Studio* 2010 telah menambahkan berbagai pembaruan dan perbaikan fitur-fitur untuk melengkapi fitur versi sebelumnya.

Framework terbaru, yaitu .Net Framework 3.5 yang merupakan pengembangan sebelumnya dari Net Framework 4.0. Database standart yang disertai dalam *Visual Studio* 2010 adalah *Microsoft SQL server* 2010 Express. (Wahana Komputer; 2013: 2).



Gambar II.2. Tampilan Microsoft Visual Studio 2010
Sumber : (Wahana Komputer ; 2013)

II.6. Pengertian *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*. Pengaplikasian *database* dapat kita lihat dan rasakan dalam keseharian kita. *Database* ini menjadi penting untuk mengelola data dari berbagai kegiatan. Misalnya, kita bisa menggunakan mesin ATM *(*anjungan tunai mandiri / *automatic teller machine*) bank karena bank telah mempunyai *database* tentang

nasabah dan rekening nasabah. Kemudian data tersebut dapat diakses melalui mesin ATM ketika bertransaksi melalui ATM. Pada saat melakukan transaksi, dalam konteks *database* sebenarnya kita sudah melakukan perubahan *(update)* data pada *database* di bank. Ketika kita menyimpan alamat dan nomor telepon di HP, sebenarnya juga telah menggunakan konsep *database*. Data yang kita simpan di HP juga mempunyai struktur yang diisi melalui formulir *(form)* yang disediakan. Pengguna dimungkinkan menambahkan nomor HP, nama pemegang, bahkan kemudian dapat ditambah dengan alamat *email*, alamat *web*, nama kantor, dan sebagainya.(Mujilan Agustinus ; 2012)

II.7. Pengertian SQL Server 2008

SQL Server 2008 adalah sebuah terobosan baru dari Microsoft dalam bidang database. SQL Server adalah sebuah *DBMS (Database Management System)* Yang dibuat oleh *Microsof*t 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. (Wenny Widya; 2010 : 3).

Gambar II.3. Tampilan SQL Server 2008
Sumber : (Wenny Widya ; 2010 : 3)

II.8. Kamus Data

Kamus data (data dictionary) dipergunakan untuk memperjelas aliran data yang digambarkan pada DFD. Kamus data adalah kumpulan daftar elemen data yang mengalir pada sistem perangkat lunak sehingga masukan (input) dan keluaran (output) dapat dipahami secara umum (memiliki standar cara penulisan). Kamus data biasanya berisi:

1. Nama - nama dari data
2. Digunakan pada – merupakan proses-proses yang terkait data
3. Deskripsi – merupakan deksripsi data
4. Informasi tambahan – seperti tipe data, nilai data, batas nilai data, dan komponen yang membentuk data. (Rosa A.S; 2011: 67).

Tabel II.1. Contoh Kamus Data

No	Nama Field	Tipe Data	Panjang
1	Id_pelanggan	CHAR	5
2	nama	VARCHAR	30
3	Alamat	VARCHAR	60
4	Telp	VARCHAR	15

(Sumber : Rosa A.S ; 2011: 67)

II.9. 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 relasional.

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 telah ditemukan.

1. Bentuk Nomal Pertama (1NF)

   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.

2. Bentuk Normal Kedua (2NF)

   Definisi bentuk normal kedua menyatakan bahwa tabel dengan kunci utama gabungan hanya dapat berada pada INF, tetapi tidak pada 2NF. Sebuah tabel relasional berada pada bentuk normal kedua jika dia berada pada 1NF dan setiap kolom bukan kunci yang sepenuhnya tergantung pada kunci utama, ini berarti bahwa setiap kolom bukan kunci harus tergantung pada seluruh kolom yang membentuk kunci utama.

3. Bentuk Normal Ketiga (3NF)

   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 utamanya. Dengan kata lain, semua atribut bukan kunci tergantung secara fungsional hanya pada kunci utama.

4. Bentuk Normal Keempat (4NF)

   Bentuk normal keempat (4NF) didasarkan pada konsep ketergantungan multivalue (MVD). Sebuah ketergantungan multivalue terjadi ketika dalam sebuah tabel relasional yang mengandung setidaknya tiga kolom, satu kolom mempunyai banyak baris bernilai sama.

5. Bentuk Normal Kelima (5NF)

   Sebuah tabel berada pada bentuk normal kelima (5NF) jika dia 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. (Janner Simarmata; 2010: 77).

II.10. Pengertian UML

UML sesuai dengan kata terakhir dari kepanjangannya, UML itu adalah salah satu bentuk *language* atau bahasa. Menurut pencetusnya, UML didefenisikan sebagai bahasa visual untuk menjelaskan, memberikan spesifikasi, membuat model, dan mendokumentasikan adpek-aspek dari sebuah sistem.

Karena tergolong bahasa visual, UML lebih mengedepankan penggunaan diagram untuk menggambarkan aspek dari sistem yang sedang dimodelkan.

Unified Modelling Language (UML) biasanya digunakan untuk:

1. Menggambarkan batasan sistem dan fungsi-fungsi sistem secara umum.
2. Menggambarkan kegiatan atau proses bisnis yang dilaksakan secara umum.
3. Menggambarkan representasi stuktur statik sebuah system dalam bentuk *class diagram*.
4. Membuat model *behavior* "yang menggambarkan sifat atau sebuah sistem.
5. Menyatakan arsitektur implementasi fisik menggunakan *component* dan *development diagram.*
6. Menyampaikan atau memperluas *fungsionalty* dengan *stereotype*. (Sugiarti Yuni ; 2013)

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.

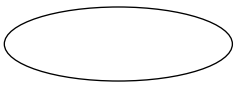
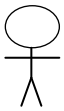
UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem.

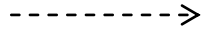
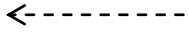
Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

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.2. Simbol *Use Case Diagram*

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.

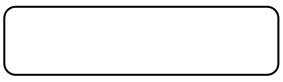
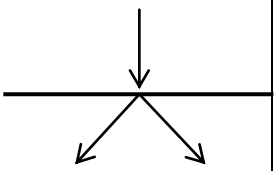
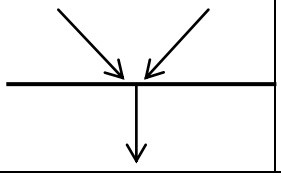
	*Include*, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	*Extend*, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

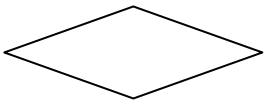
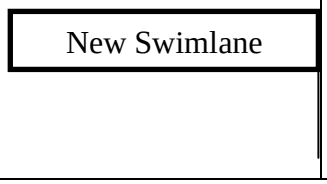
(Sumber : Windu Gata ; 2013 : 4)

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.

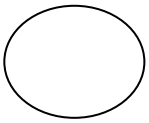
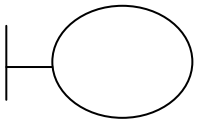
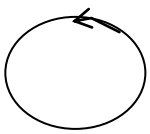
	*Decision Points,* menggambarkan pilihan untuk pengambilan keputusan, *true, false.*
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

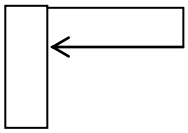
(Sumber : Windu Gata ; 2013 : 6)

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*.

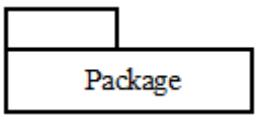
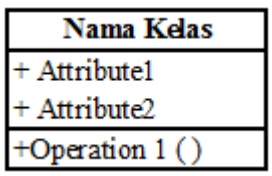
	*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation*, *activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

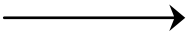
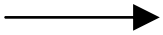
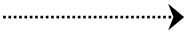
(Sumber : Windu Gata ; 2013 : 7)

4. Diagram Kelas (*Class Diagram*)

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

Tabel II.5. Simbol *Class Diagram*

Simbol	**Deskripsi**
Package 	Package merupakan bungkusan dari satu atau lebih kelas
Operasi 	Kelas pada struktur system

Antarmuka/interface  interface	Sama dengan konsep *interface* dalam pemograman berorientasi objek
Asosiasi 1 1..*	Relasi antar kelas dengan maknaumum, asosiasi biasanya juga disertai dengan *multiplicity*
Asosiasi berarah / directed asosiasi 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
Generalisasi 	Relasi antar kelas dengan makna *generalisasi-spesialisasi* (umum-khusus)
Kebergantungan (*defedency)* 	Relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi 	Relasi antar kelas dengan makna semua-bagian (*whole-part*)

(Sumber: Yuni Sugiarti ; 2013 : 59)

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.6. ***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 : 8)