BAB II

LANDASAN TEORI

II.1. Kecerdasan Buatan

Kecerdasan buatan berasal dari bahasa Inggris *"Artificial Intelligence"* atau disingkat AI, yaitu *intelligence* adalah kata sifat yang berarti cerdas, sedangkan *artificial* artinya buatan. Kecerdasan buatan yang dimaksud di sini merujuk pada mesin yang mampu berpikir, menimbang tindakan yang akan diambil, dan mampu mengambil keputusan seperti yang dilakukan oleh manusia berikut adalah beberapa definisi kecerdasan buatan yang telah didefinisikan oleh beberapa ahli.

Alan Turing, ahli matematika berkebangsaan Inggris yang dijuluki bapak komputer modern dan pembongkar sandi Nazi dalam era Perang Dunia II 1950, menetapkan definisi *Artificial Intelligent :* "Jika komputer tidak dapat dibedakan dengan manusia saat berbincang mealui terminal komputer, maka bisa dikatakan komputer itu cerdas, mempunyai kecerdasan".

Herbert Alexander Simon, "Kecerdasan buatan (*Artificial Intelligence*) merupakan kawasan penelitian, aplikasi, dan instruksi yang terkait dengan pemrograman komputer untuk melakukan sesuatu hal yang dalam pandangan manusia adalah cerdas.

Menurut Winston dan Prendergast, tujuan dari kecerdasan buatan adalah :

1. Membuat mesin menjadi lebih pintar (tujuan utama)
2. Memahami apa itu kecerdasan (tujuan ilmiah)

3. Membuat mesin lebih bermanfaat (tujuan *entrepreneurial)*

Berdasarkan definisi ini, maka kecerdasan buatan menawarkan media maupun uji teori tentang keccerdasan. Teori-teori ini nantinya dapat dinyatakan dalam bahasa pemrograman dan eksekusinya dapat dibuktikan pada komputer nyata. (T. Sutojo, dkk ; 2011 : 1-3)

II.2. Sistem Pakar

Sistem pakar adalah suatu sistem yang dirancang untuk dapat menirukan keahlian seorang pakar dalam menjawab pertanyaan dan memecahkan masalah. Sistem pakar akan memberikan pemecahan suatu masalah yang didapat dari dialog dengan pengguna. Dengan bantuan Sistem Pakar seseorang yang bukan pakar/ahli dapat menjawab pertanyaan, menyelesaikan masalah serta mengambil keputusan yang biasanya dilakukan oleh seorang pakar. (T. Sutojo, dkk ; 2011 : 13)

Istilah sistem pakar berasal dari istilah *knowledge-based expert system*. Istilah ini muncul karena untuk memecahkan masalah, sistem pakar menggunakan pengetahuan seorang pakar yang dimasukkan ke dalam komputer. Seseorang yang bukan pakar menggunakan sistem pakar untuk meningkatkan kemampuan pemecahan masalah, sedangkan seorang pakar menggunakan sistem pakar untuk *knowledge assistant*. Berikut adalah beberapa pengertian sistem pakar. (T. Sutojo, dkk ; 2011 : 160)

1. Turban

   “Sistem pakar adalah sebuah sistem yang menggunakan pengetahuan manusia di mana pengetahuan tersebut dimasukkan ke dalam sebuah komputer dan kemudian digunakan untuk menyelesaikan masalah-masalah yang biasanya membutuhkan kepakaran atau keahlian manusia”.

2. Jackson

   “Sistem pakar adalah program komputer yang merepresentasikan dan melakukan penalaran dengan pengetahuan beberapa pakar untuk memecahkan masalah atau memberikan saran”.

3. Luger dan Stubblefield

   “Sistem pakar adalah program yang berbasiskan pengetahuan yang menyediakan solusi ‘kualitas pakar’ kepadda masalah-masalah dalam bidang (domain) yang spesifik”.

II.2.1. Manfaat Sistem Pakar

Sistem pakar menjadi sangat populer karena sangat banyak kemampuan dan manfaat yang diberikannya, di antaranya (T. Sutojo, dkk ; 2011 : 160-161) :

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. 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 pelatihan. Pengguna pemula yang bekerja dengan sistem pakar akan menjadi lebih berpengalaman karena adanya fasilitas penjelas yang berfungsi sebagai guru.
11. Meningkatkan kemampuan untuk menyelesaikan masalah karena sistem pakar mengambil sumber pengetahuan dari banyak pakar.

II.2.2. Ciri-Ciri Sistem Pakar

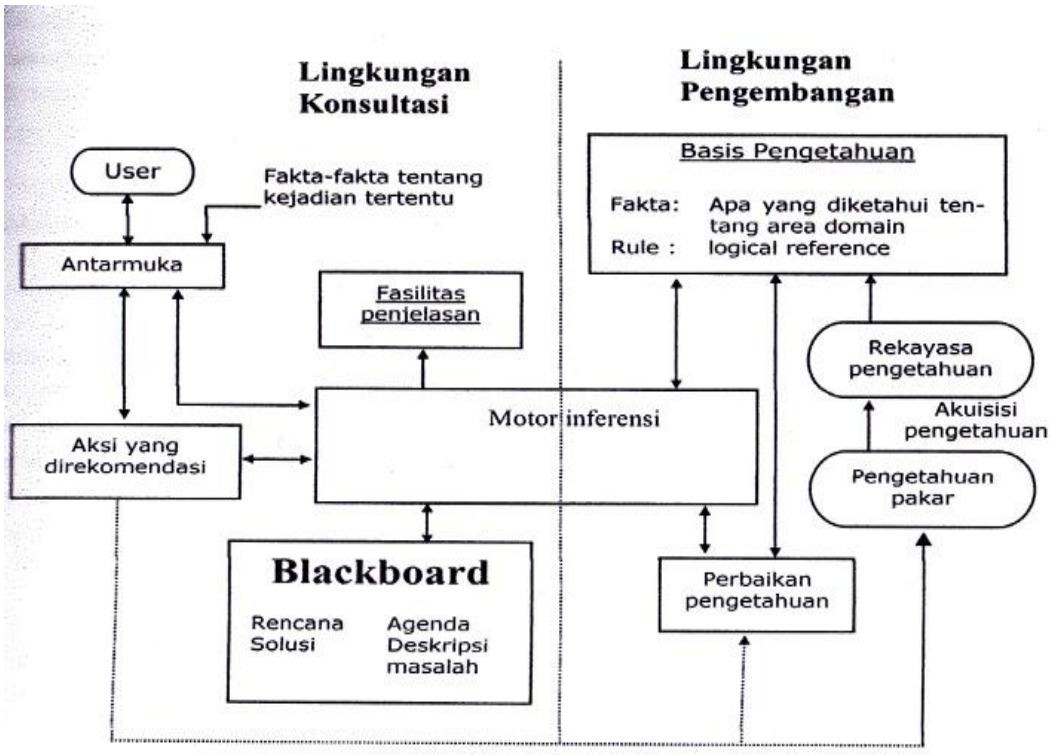
Ciri-ciri dari sistem pakar adalah sebagai berikut (T. Sutojo, dkk ; 2011 : 162) :

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

Menurut T. Sutojo, dkk (2011 : 166), ada 2 bagian penting dari sistem pakar, yaitu lingkungan konsultasi (*consultation environment*). Lingkungan pengembangan digunakan oleh pembuat sistem pakar untuk membangun komponen-komponennya dan memperkenalkan pengetahun kedalam *knowledge base* (basis pengetahuan). Lingkungan konsultasi digunakan oleh pengguna untuk berkonsultasi sehingga pengguna mendapatkan pengetahuan dan nasehat dari sistem pakar layaknya berkonsultasi dengan seorang pakar. Adapun komponen-komponen yang penting dalam sebuah sistem pakar dapat dilihat pada gambar II.4 berikut ini :



Gambar II.1. Komponen-komponen yang penting dalam sebuah sistem pakar

(Sumber : T. Sutojo, dkk ; 2011 : 167)

Penjelasan tentang gambar II.1 adalah sebagai berikut (T. Sutojo, dkk ; 2011 : 167-169) :

1. Akuisisi pengetahuan

   Susbsitem ini digunakan unutk memasukkan pengetahuan dari seorang pakar dengan cara merekayasa pengetahuan agar bisa diproses oleh komputer dan menaruhnya kedalam basis pengetahuan dengan format tertentu. Sumber-sumber pengetahuan bisa diambil 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.

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 dan kesimpulan.

4. Daerah Kerja (*Blackboard*)

   Untuk merekam hasi sementara yang akan dijadikan sebagai keputusan dan untuk menjelaskan sebuah masalah yang akan terjadi, sistem pakar membutuhkan *Blackboard*, yaitu area pada memori yang berfungsi sebagai basis data.

5. Antarmuka Pengguna (*User Interface*)

   Digunakan sebagai media komunikasi anatar pengguna dan sistem pakar. Komunikasi ini yang paling bagus bila disajikan dalam bahasa alami (*natural language*) dan dilengkapi dengan grafik, menu, dan formulir elektronik.

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

   Berfungsi memberikan penjelasan kepada pengguna, bagaimana atau kesimpulan dapat diambil. Kemampuan seperti saat 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 pegetahuan, belajar dari kesalahan masa lalu, kemudian memperbaiki pengetahuan sehingga dapat dipakai di massa mendatang. Kemampuan evaluasi siri seperti itu diperlukan oleh program agar dapat menganalisis alasan-alasan kesuksesan dan kegagalan dalam mengambil kesimpulan. Denga cara ini basis pengetahuan yang lebih baik dan penalaran yang leih efektif akan dihasilkan.

8. Pengguna (*User*)

   Pada umumnya pengguna sistem pakar bukanlah seorang pakar (*non-expert*) yang membutuhkan solusi, saran, atau pelatihan (training) dari berbagai permasalahan yang ada.

II.3. Metode Teorema Bayes

Teori Bayes dikemukakan oleh seorang pendeta Inggris pada tahun 1763 yang bernama Thomas Bayes. Teori Bayes ini kemudian disempurnakan oleh Laplace. Teori Bayes digunakan untuk menghitung probabilitas terjadinya suatu peristiwa berdasarkan pengaruh yang didapat dari hasil observasi. Teori Bayes merupakan kaidah yang memperbaiki atau merevisi suatu probabilitas dengan cara memanfaatkan informasi tambahan. Maksudnya, dari probabilitas awal (*prior*

probability) yang belum diperbaiki yang dirumuskan berdasarkan informasi yang tersedia saat ini, kemudian dibentuklah probabilitas berikutnya (*posterior probability*) (Hartatik dan Ketut ; 2015).

Bentuk teorema Bayes untuk *evidence* tunggal E dan hipotesis tunggal H adalah (T. Sutojo, dkk ; 2011 : 189-191) :

$$P(H|E) = \frac{P(E|H) \times P(H)}{P(E)} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (II.1)$$

dengan:

P(H | E) = probabilitas hipotesis H terjadi jika *evidence* E terjadi

P(E | H) = probailitas munculnya *evidence* E, jika hipotesis H terjadi

P(H) = probabilitas hipotesis H tanpa memandang *evidence* apa pun

P(E) = probabilitas *evidence* E tanpa memandang apa pun

Bentuk teorema Bayes untuk *evidence* tunggal E dan hipotesis ganda H_1, H_2,...., H_n adalah:

$$P(H_i|E) = \frac{P(E|H_i) \times P(H_i)}{\sum_{k=1}^{n} P(E|H_k) \times P(H_k)} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (II.2)$$

dengan:

$P(H_i | E)$ = probabilitas hipotesis H_i terjadi jika *evidence* E terjadi

$P(E | H_i)$ = probailitas munculnya *evidence* E, jika hipotesis H_i terjadi

$P(H_i)$ = probabilitas hipotesis H_i tanpa memandang *evidence* apa pun

n = jumlah hipotesis yang terjadi

Untuk *evidence* ganda E_1, E_2,....., E_m dan hipotesis ganda H_1, H_2,...., H_n adalah:

$$P(H_i \mid E_1 E_2 \ldots . E_m) = \frac{P(E_1 E_2 \ldots . E_m \mid H_i) \times P(H_i)}{\sum_{k=1}^{n} P(E_1 E_2 \ldots . E_m \mid H_k) \times P(H_k)} \ldots\ldots\ldots\ldots\ldots (II.3)$$

Untuk mengaplikasikan persamaan (3), maka harus diketahui probabilitas bersyarat dari semua kombinasi yang mungkin dari *evidence-evidence* untuk seluruh hipotesis. Secara praktis, hal ini tidak mungkin bisa dilakukan. Oleh karena itu, persamaan (3) diganti dengan persamaan (4):

$$P(H_i \mid E_1 E_2 \ldots E_m) = \frac{P(E_1 \mid H_i) \times P(E_2 \mid H_i) \times \ldots \times P(E_m \mid H_i) \times P(H_i)}{\sum_{k=1}^{n} P(E_1 \mid H_k) \times P(E_2 \mid H_k) \times \ldots \times P(E_m \mid H_k) \times P(H_k)} . (II.4)$$

II.4. Pengertian Basis Data

Basis data dapat dipahami sebagai suatu kumpulan data terhubung (*interrelated data*) yang disimpan secara bersama-sama pada suatu media, tanpa *mengatap* satu sama lain atau tidak perlu suatu kerangkapan data (kalaupun ada maka kerangkapan data tersebut harus seminimal mungkin dan terkontrol [*controlled redudancy*]), data disimpan dengan cara-cara tertentu sehingga mudah digunakan/atau ditampilkan kembali; data dapat digunakan oleh satu atau lebih program-program aplikasi secara optimal; data disimpan tanpa mengalami ketergantungan dengan program yang akan menggunakannya; data disimpan sedemikian rupa sehingga proses penambahan, pengambilan, dan modifikasi data dapat dilakukan dengan mudah dan terkontrol (Sutanta ; 2011 : 29-30).

II.5. Normalisasi

Normalisasi diartikan sebagai suatu teknik yang menstrukurkan/ mendekomposisi data dalam cara-cara tertentu untuk mencegah timbulnya

permasalahan pengolahan data dalam basis data. Permasalahan yang dimaksud adalah berkaitan dengan penyimpangan-penyimpangan *(anomalies)* yang terjadi akibat adanya kerangkapan data dalam relasi dan in-efisiensi pengolahan (Martin, 1975). (Sutanta ; 2011 : 174)

Proses normalisasi menghasilkan relasi yang optimal, yaitu (Martin, 1975) : (Sutanta ; 2011 : 175)

1. Memiliki struktur *record* yang konsisten secara logik;
2. Memiliki struktur *record* yang mudah untuk dimengerti;
3. Memiliki struktur *record* yang sederhana dalam pemeliharaan;
4. Memiliki struktur *record* yang mudah ditampilkan kembali untuk memenuhi kebutuhan pengguna;
5. Minimalisasi kerangkapan data guna meningkatkan kinerja sistem.

Secara berturut-turut masing-masing level normal tersebut dibahas berikut ini, dimulai dari bentuk tidak normal. (Sutanta ; 2011 : 176-179)

1. Relasi bentuk tidak normal (*Un Normalized Form* / UNF)

   Relasi-relasi yang dirancang tanpa mengindahkan batasan dalam defisi basis data dan karakteristik *Relational Database Management System* (RDBM) menghasilkan relasi *Un Normalized Form* (UNF). Bentuk ini harus di hindari dalam perancangan relasi dalam basis data. Relasi *Un Normalized Form* (UNF) mempunyai kriteria sebagai berikut.

a. Jika relasi mempunyai bentuk *non flat file* (dapat terjadi akibat data disimpan sesuai dengan kedatangannya, tidak memiliki struktur tertentu, terjadi duplikasi atau tidak lengkap)
b. Jika relasi membuat *set atribut* berulang *(non single values)*
c. Jika relasi membuat *atribut non atomic value*

2. Relasi bentuk normal pertama (*First Norm Form* / 1NF)

   Relasi disebut juga *First Norm Form* (1NF) jika memenuhi kriteria sebagai berikut.

   a. Jika seluruh atribut dalam relasi bernilai *atomic (atomic value)*
   b. Jika seluruh atribut dalam relasi bernilai tunggal *(single value)*
   c. Jika relasi tidak memuat set atribut berulang
   d. Jika semua record mempunyai sejumlah atribut yang sama.

   Permasalahan dalam *First Norm Form* (1NF) adalah sebagai berikut.

   a. Tidak dapat menyisipkan informasi parsial
   b. Terhapusnya informasi ketika menghapus sebuah *record*

3. Bentuk normal kedua *(Second Normal Form /* 2NF)

   Relasi disebut sebagai *Second Normal Form (*2NF) jika memenuhi kriteria sebagai berikut

   a. Jika memenuhi kriteria *First Norm Form (*1NF)
   b. Jika semua atribut nonkunci *Functional Dependence* (FD) pada *Primary Key* (PK)

   Permasalahan dalam *Second Normal Form /* 2NF adalah sebagai berikut

a. Kerangkapan data *(data redundancy)*
b. Pembaharuan yang tidak benar dapat menimbulkan inkonsistensi data *(data inconsistency)*
c. Proses pembaharuan data tidak efisien

Kriteria tersebut mengidentifikasikan bahwa antara atribut dalam *Second Normal Form* masih mungkin mengalami *Third Norm Form*. Selain itu, relasi *Second Normal Form* (2NF) menuntut telah didefinisikan atribut *Primary Key* (PK) dalam relasi. Mengubah relasi *First Norm Form* (1NF) menjadi bentuk *Second Normal Form* (2NF) dapat dilakukan dengan mengubah struktur relasi dengan cara :

a. Identifikasikan *Functional Dependence* (FD) relasi *First Norm Form* (1NF)
b. Berdasarkan informasi tersebut, dekomposisi relasi *First Norm Form* (1NF) menjadi relasi-relasi baru sesuai *Functional Dependence* nya. Jika menggunakan diagram maka simpul-simpul yang berada pada puncak diagram ketergantungan data bertindak *Primary Key* (PK) pada relasi baru

4. Bentuk normal ketiga (*Third Norm Form* / 3NF)

Suatu relasi disebut sebagai *Third Norm Form* jika memenuhi kriteria sebagai berikut.

a. Jika memenuhi kriteria *Second Normal Form* (2NF)
b. Jika setiap atribut nonkunci tidak (TDF) *(Non Transitive Dependeny)* terhadap *Primary Key* (PK)

Permasalahan dalam *Third Norm Form* (3NF) adalah keberadaan penentu yang tidak merupakan bagian dari *Primary Key* (PK) menghasilkan duplikasi rinci data pada atribut yang berfungi sebagai *Foreign Key* (FK) (duplikasi berbeda dengan keterangan data).

Mengubah relasi *Second Normal Form* (2NF) menjadi bentuk *Third Norm Form* (3NF) dapat dilakukan dengan mengubah struktur relasi dengan cara :

a. Identifikasi TDF relasi *Second Normal Form* (2NF)
b. Berdasarkan informasi tersebut, dekomposisi relasi *Second Normal Form* (2NF) menjadi relasi-relasi baru sesuai TDF-nya.

5. Bentuk normal *Boyce-Cood (Boyce-Codd Norm Form* / BCNF)

   Bentuk normal *Boyce-Codd Norm Form* (BCNF) dikemukakan oleh R.F. Boyce dan E.F. Codd. Suatu relasi disebut sebagai *Boyce-Codd Norm Form* (BCNF) jika memenuhi kriteria sebagai berikut.

   a. Jika memenuhi kriteria *Third Norm Form* (3NF)
   b. Jika semua atribut penentu (determinan) merupakan CK

6. Bentuk normal keempat (*Forth Norm Form* / 4NF)

   Relasi disebut sebagi *Forth Norm Form* (4NF) jika memenuhi kriteria sebagai berikut.

   a. Jika memenuhi kriteria *Boyce-Codd Norm Form*.
   b. Jika setiap atribut didalamnya tidak mengalami ketergantungan pada banyak nilai.

7. Bentuk normal kelima (*Fifth Norm Form* / 5NF)

Suatu relasi memenuhi kriteria *Fifth Norm Form* (5NF) jika kerelasian antar data dalam relasi tersebut tidak dapat direkonstruksi dari struktur relasi yang sederhana.

8. Bentuk normal kunci domain (*Domain Key Norm Form* / DKNF)

   Relasi disebut sebagai *Domain Key Norm Form* (DKNF) jika setiap batasan dapat disimpulkan secara sederhana dengan mengetahui sekumpulan nama atribut dan domainnya selama menggunkan sekumpulan atribut pada kuncinya.

II.6. *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah sebuah bahasa yang diterima dan digunakan oleh *software developer* dan *software analyst* sebagai suatu bahasa yang cocok untuk merepresentasikan grafik dari suatu relasi antar entitas-entitas software (Gornik, 2003). Dengan menggunakan UML, tim pengembang *software* akan mempunyai banyak keuntungan, seperti memudahkan komunikasi dengan sesama anggota tim tentang *software* apa yang akan dibuat, memudahkan integrasi ke dalam area pengerjaan *software* karena bahasa ini berbasiskan *meta-models* dimana *meta-models* bisa mendefinisikan proses-proses untuk mengkonstruksikan konsep-konsep yang ada. UML juga menggunakan formatur *input* dan *output* yang sudah mempunyai bentuk standar yaitu *XML Metadata Interchange* (XMI), menggunakan aplikasi dan pemodelan data yang universal, merepresentasikan dari tahap analisis ke implementasi lalu ke *deployment* yang terpadu, dan mendeskripsikan keutuhan tentang spesifikasi software.

UML menyediakan kumpulan alat yang sudah terstandarisasi, yang digunakan untuk mendokumentasikan analisis dan perancangan sebuah sistem perangkat lunak. (Kendall & Kendall, 2005, p663) Peralatan utama UML adalah diagram-diagram yang digunakan untuk membantu manusia dalam memvisualisasikan proses pengembangan sebuah sistem perangkat lunak, sama seperti penggunaan denah (*blueprint*) dalam pembuatan bangunan (Winata dan Setiawan ; 2013).

II.6.1. Tujuan Pemanfaatan *Unified Modeling Language* (UML)

Tujuan dari penggunaan diagram seperti diungkapan oleh Schmuller J. (2004), *"The purpose of the diagrams is to present multiple views of a system; this set of multiple views is called a model".*

Berikut tujuan utama dalam desain UML adalah (Sugrue J. 2009) : (Haviluddin ; 2011 : 2).

1. Menyediakan bagi pengguna (analisis dan desain sistem) suatu bahasa pemodelan visual yang ekspresif sehingga mereka dapat mengembangkan dan melakukan pertukaran model data yang bermakna.
2. Menyediakan mekanisme yang spesialisasi untuk memperluas konsep inti.
3. Karena merupakan bahasa pemodelan visual dalam proses pembangunannya maka UML bersifat independen terhadap bahasa pemrograman tertentu.

4. Memberikan dasar formal untuk pemahaman bahasa pemodelan.
5. Mendorong pertumbuhan pasar terhadap penggunaan alat desain sistem yang berorientasi objek (OO).
6. Mendukung konsep pembangunan tingkat yang lebih tinggi seperti kolaborasi, kerangka, pola dan komponen terhadap suatu sistem.
7. Memiliki integrasi praktik terbaik.

Ada 4 (empat) prinsip dasar dari pemrograman berorientasi obyek yang menjadi dasar kemunculan UML, yaitu *abstraksi, enkapsulasi, modularitas* dan *hirarki*. Berikut dijelaskan satu persatu secara singkat.

1. *Abstraksi* memfokuskan perhatian pada karakteristik obyek yang paling penting dan paling dominan yang bisa digunakan untuk membedakan obyek tersebut dari obyek lainnya.
2. *Enkapsulasi* menyembunyikan banyak hal yang terdapat dalam obyek yang tidak perlu diketahui oleh obyek lain. Dalam praktek pemrograman, enkapsulasi diwujudkan dengan membuat suatu kelas interface yang akan dipanggil oleh obyek lain, sementara didalam obyek yang dipanggil terdapat kelas lain yang mengimplementasikan apa yang terdapat dalam kelas *interface*.
3. *Modularitas* membagi sistem yang rumit menjadi bagian-bagian yang lebih kecil yang bisa mempermudah developer memahami dan mengelola obyek tersebut.
4. *Hirarki* berhubungan dengan abstraksi dan modularitas, yaitu pembagian berdasarkan urutan dan pengelompokkan tertentu.

Misalnya untuk menentukan obyek mana yang berada pada kelompok yang sama, obyek mana yang merupakan komponen dari obyek yang memiliki hirarki lebih tinggi. Semakin rendah hirarki obyek berarti semakin jauh abstraksi dilakukan terhadap suatu obyek.

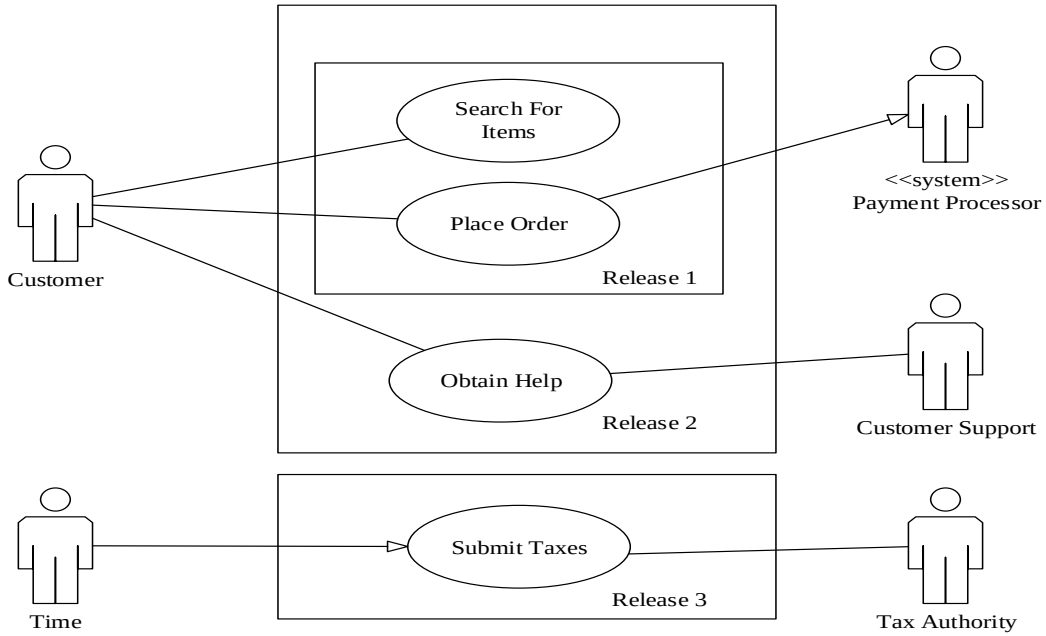
II.6.2. Diagram-Diagram *Unified Modeling Language* (UML)

Adapun jenis-jenis dari diagram UML adalah sebagai berikut :

1. *Use Case* Diagram

   Diagram yang menggambarkan *actor, use case* dan relasinya sebagai suatu urutan tindakan yang memberikan nilai terukur untuk aktor. Sebuah *use case* digambarkan sebagai *elips horizontal* dalam suatu diagram UML *use case*. *Use Case* memiliki dua istilah, yaitu :

   a. *System use case*; interaksi dengan sistem.
   b. *Business use case*; interaksi bisnis dengan konsumen atau kejadian nyata.

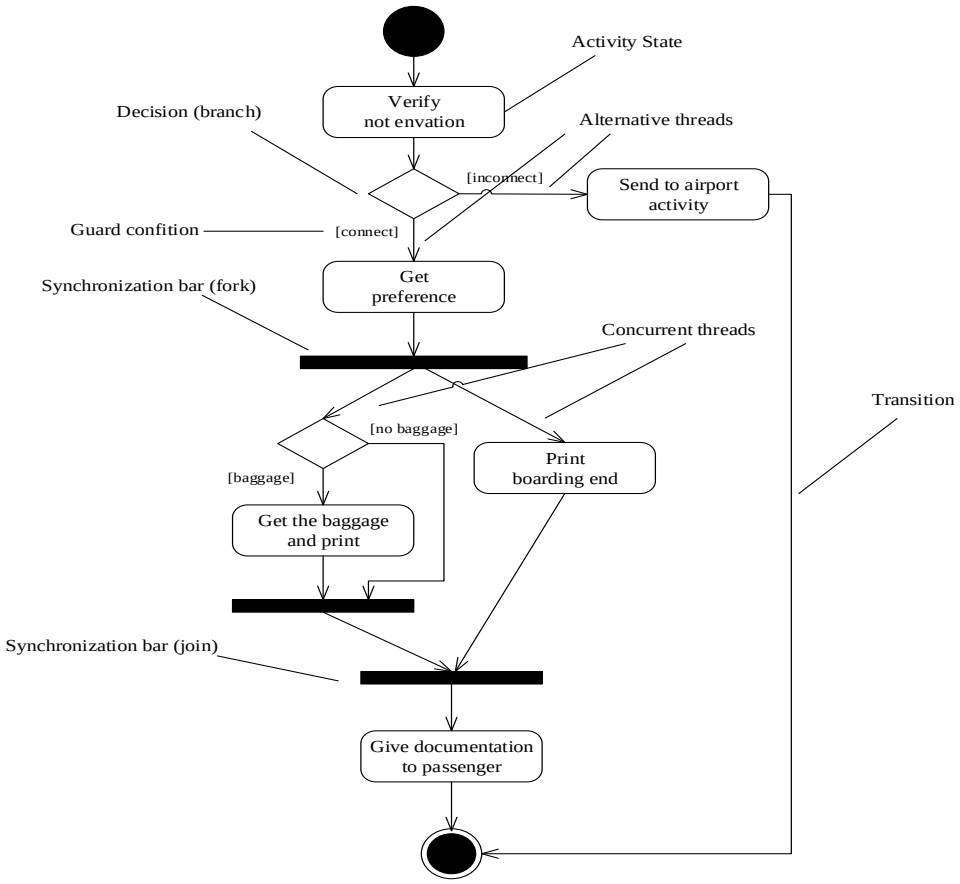


Gambar II.2. Notasi *Use Case* Diagram

(Sumber : Haviluddin ; 2011)

2. *Activity* Diagram

   Menggambarkan aktifitas-aktifitas, objek, *state*, *transisi state* dan *event*. Dengan kata lain kegiatan diagram alur kerja menggambarkan perilaku sistem untuk aktivitas. Berikut notasi *activity diagram* dapat dilihat pada Gambar II.3. di bawah ini.



Gambar II.3. Notasi *Activity* Diagram

(Sumber : Haviluddin ; 2011)

3. *Class* Diagram

Class diagram menggambarkan struktur statis dari kelas dalam sistem anda dan menggambarkan atribut, operasi dan hubungan antara kelas. *Class diagram* membantu dalam memvisualisasikan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. Selama tahap desain, *class diagram* berperan dalam

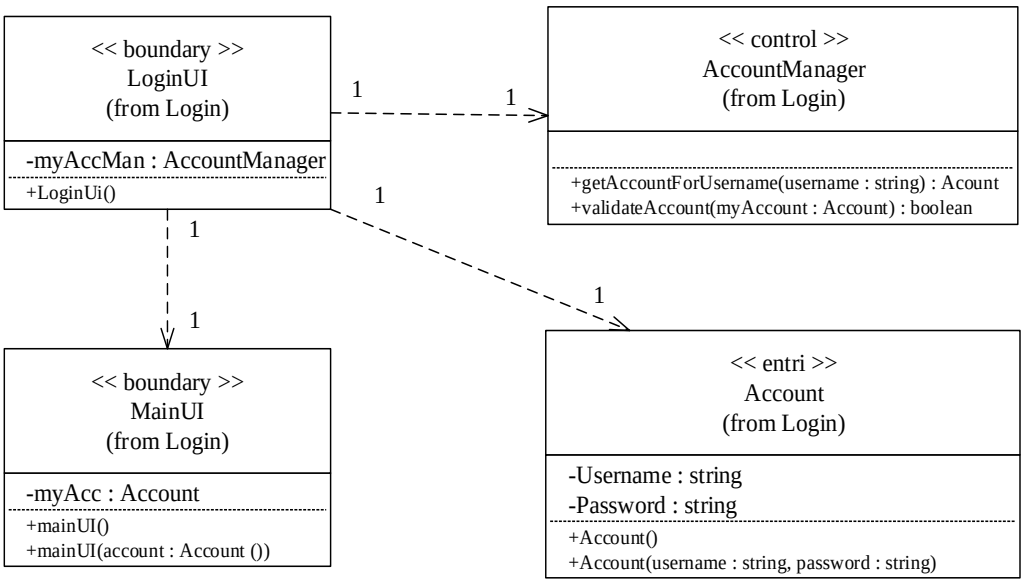
menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat.

Class memiliki tiga area pokok :

a. Nama (dan *stereotype*)

b. Atribut

c. Metoda.



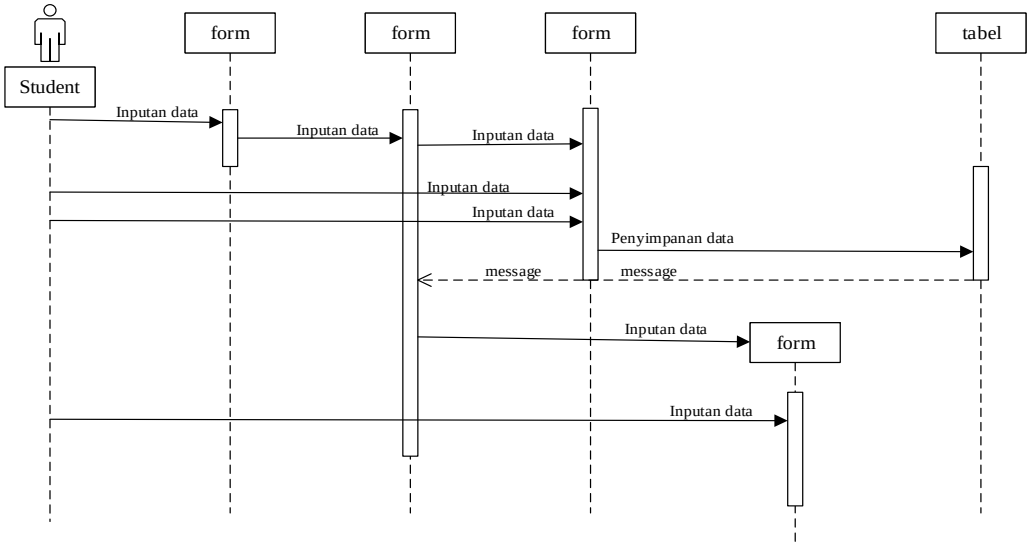
Gambar II.4. Notasi *Class* Diagram

(Sumber : Haviluddin ; 2011)

4. *Sequence* Diagram

Sequence diagram menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. Secara mudahnya *sequence diagram* adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan *use case diagram*.

.



G

ambar II.5. Notasi *Sequence* Diagram

(Sumber : Haviluddin ; 2011)

II.7. PHP (*Hypertext Preprocessor)*

PHP adalah bahasa *server-side scripting* yang menyatu dengan HTML untuk membuat halaman web yang dinamis. Karena PHP merupakan *server-side scripting* maka sintaks dan perintah-perintah PHP akan dieksekusi di *server* kemudian hasilnya dikirimkan ke *web browser* dalam format HTML. Dengan demikian kode program yang ditulis dalam PHP tidak akan terlihat oleh *user* sehingga keamanan halaman web lebih terjamin. PHP dirancang untuk membentuk halaman web dinamis, yaitu halaman web yang dapat membentuk suatu tampilan berdasarkan permintaan terkini, seperti menampilkan isi basis data ke halaman web.

PHP termasuk dalam *Open Source Product,* sehingga *source code* PHP dapat diubah dan didistribusikan secara bebas. PHP juga dapat berjalan pada berbagai *web server* seperti ISS (*Internet Information Server*), PWS (*Personal Web Server*), Apache, Xitami. PHP juga mampu lintas *platform.* Artinya PHP dapat berjalan di banyak sistem operasi yang beredar saat ini, di antaranya: Sistem Operasi Microsoft Windows (semua versi), Linux, Mac OS, Solaris. PHP dapat dibangun sebagai modul pada *web server* Apache dan sebagai *binary* yang dapat berjalan sebagai CGI (*Common Gateway Interface).* PHP dapat mengirim HTTP *header*, dapat mengatur *cookies,* mengatur *outhentication* dan *redirect users*.

Salah satu keunggulan yang dimiliki oleh PHP adalah kemampuannya untuk melakukan koneksi ke berbagai macam *software* sistem manajemen basis data/*Database Management System* (DBMS), sehingga dapat menciptakan suatu halaman web yang dinamis. PHP mempunyai koneksitas yang baik dengan beberapa DBMS antara lain Oracle, Sybase, mSql, MySQL, Microsoft SQL Server, Solid, PostgreSQL, Adabas, FilePro, Velocis, dBase, Unix dbm, dan tak terkecuali semua database ber-*interface* ODBC (Arief ; 2011 : 43).

II.8. MySQL

MySQL adalah salah satu jenis *database server* yang sangat terkenal dan banyak digunakan untuk membangun aplikasi web yang menggunakan *database* sebagai sumber dan pengolahan datanya. Kepopuleran MySQL antara lain karena MySQL menggunakan SQL sebagai bahasa dasar untuk mengakses *database*-nya sehingga mudah untuk digunakan, kinerja *query* cepat, dan mencukupi untuk

kebutuhan *database* perusahaan-perusahaan skala menengah-kecil. MySQL juga bersifat *open source* dan *free* pada berbagai *platform* (kecuali pada windows yang bersifat *shareware*). MySQL didistribusikan dengan lisensi *open source* GPL (*General Public License*) mulai versi 3.23, pada bulan Juni 2000.

MySQL merupakan *database* yang pertama kali didukung oleh bahasa pemrograman *script* untuk internet (PHP dan Perl). MySQL dan PHP dianggap sebagai pasangan *software* pengembangan aplikasi web yang ideal. MySQL lebih sering digunakan untuk membangun aplikasi berbasis web, umumnya pengembangan aplikasinya menggunakan bahasa pemrograman *script* PHP (Arief ; 2011 : 151).