BAB 2
LANDASAN TEORI

2.1 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 yang dimaksud disini merajuk pada mesin yang mampu berpikir, menimbang tindakan yang diambil, dan mampu mengambil keputusan seperti yang dilakukan oleh manusia.

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

Menurut John Carthy dari Stanford mendefinisikan kecerdasan sebagai "Kemampuan untuk mencapai sukses dalam menyelesaikan suatu permasalahan". Menurut Herbert Alexander Simon (June 15, 1916-February 9, 2001): "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 Rich and Knight (1991): "Kecerdasan buatan (AI) merupakan sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia".

Dari sini dapat dikatakan bahwa: cerdas adalah memiliki pengetahuan, pengalaman, dan penalaran untuk membuat keputusan dan mengambil tindakan. Jadi, agar mesin bisa cerdas (bertindak seperti manusia) maka harus diberi bekal pengetahuan dan diberi kemampuan untuk menalar. [2]

2.1.1 Keuntungan Kecerdasan Buatan

Beberapa keuntungan dalam kecerdasan buatan yaitu: [3]

1. Kecerdasan buatan lebih bersifat permanen. Kecerdasan alami akan cepat mengalami perubahan. Hal ini dimungkinkan karena sifat manusia yang pelupa. Kecerdasan buatan tidak akan berubah sepanjang sistem komputer dan program tidak mengubahnya.
2. Kecerdasan buatan lebih mudah diduplikasi dan disebarkan. Men-*transfer* pengetahuan manusia dari satu orang ke orang lain butuh proses dan waktu lama. Disamping itu suatu keahlian tidak akan pernah bisa diduplikasi secara lengkap. Sedangkan jika pengetahuan terletak pada suatu sistem komputer, pengetahuan tersebuat dapat di-*transfer* atau disalin dengan mudah dan cepat dari satu komputer ke komputer lain
3. Kecerdasan buatan lebih murah dibanding dengan kecerdasan alami. Menyediakan layanan komputer akan lebih mudah dan lebih murah dibanding dengan harus mendatangkan seseorang untuk mengerjakan sejumlah pekerjaan dalam jangka waktu yang sangat lama.
4. Kecerdasan buatan bersifat konsisten. Hal ini disebabkan karena kecerdasan buatan adalah bagian dari teknologi komputer. Sedangkan kecerdasan alami senantiasa berubah-ubah.
5. Kecerdasan buatan dapat didokumentasikan. Keputusan yang dibuat komputer dapat didokumentasikan dengan mudah dengan melacak setiap aktivitas dari sistem tersebut. Kecerdasan alami sangat sulit untuk direproduksi.
6. Kecerdasan buatan dapat mengerjakan pekerjaan lebih cepat dibanding dengan kecerdasan alami.
7. Kecerdasan buatan dapat mengerjakan pekerjaan lebih baik dibanding dengan kecerdasan alami.

2.1.2 Lingkup Utama Kecerdasan Buatan

Lingkup utama kecerdasan buatan diantaranya: [2]

1. Sistem pakar. Komputer digunakan sebagai saran untuk menyimpan pengetahuan para pakar. Dengan demikian komputer akan memiliki keahlian untuk menyelesaikan masalah dengan meniru keahlian yang dimiliki para pakar

2. Pengolahan bahasa alami. Dengan pengolahan bahasa alami ini diharapkan user mampu berkomunikasi dengan komputer dengan menggunakan bahasa sehari-hari.
3. Pengenalan ucapan. Melalui pengenalan ucapan diharapkan manusia mampu berkomunikasi dengan komputer dengan menggunakan suara.
4. Robotika dan sistem sensor
5. *Computer Vision*, mencoba untuk dapat mengintrepetasikan gambar atau objek-objek tampak melalui komputer
6. *Intelligent Computer Aid Instruction*. Komputer dapat digunakan sebagai tutor yang dapat melatih dan mengajar

2.1.3 Penerapan Kecerdasan Buatan

Kecerdasan buatan dapat diterapkan pada beberapa bidang yang diantaranya: [2]

a. Visualisasi komputer

   Kecerdasan buatan pada bidang visualisasi komputer ini memungkinkan sebuah sistem komputer mengenali gambar sebagai input. Contohnya mengenali sebuah pola pada suatu gambar.

b. Pengenalan Suara

   Kecerdasan buatan pada pengenalan suara ini dapat mengenali suara manusia. Cara mengenali suara ini dengan mencocokannya pada acuan yang telah diprogramkan terlebih dahulu. Contohnya perintah komputer dengan menggunakan suara user.

c. Sistem Pakar

   Kecerdasan buatan pada sistem pakar ini memungkinkan sebuah sistem komputer memiliki cara berpikir dan penalaran seorang ahli dalam mengambil keputusan, untuk memecahkan masalah yang ada pada saat itu. Contohnya program komputer yang dapat mendiagnosa penyakit dengan memasukan gejala-gejala yang dialami pasien.

d. Permainan Kecerdasan

buatan pada permainan ini memungkinkan sebuah sistem komputer untuk memiliki cara berpikir manusia dalam bermain. Contohnya permainan yang memiliki fasilitas orang melawan komputer. Komputer sudah di program sedemikian rupa agar memiliki cara bermain seperti seorang manusia bahkan bisa melebihi seorang manusia.

2.1.4 Kecerdasan Buatan Pada *Game*

Kecerdasan buatan atau AI merupakan kegiatan membuat komputer agar dapat berpikir dan mengerjakan kegiatan yang dapat dilakukan oleh manusia maupun binatang.

Saat ini dapat ditemukan program komputer yang memiliki kemampuan menangani masalah seperti aritmatik, *sorting*, *searching*. Bahkan komputer juga dapat bermain beberapa *board game* seperti catur dan reversi lebih baik daripada manusia.

Namun masih banyak hal yang tidak dapat dilakukan dengan baik oleh komputer. Seperti, mengenali wajah, berbicara bahasa manusia, menentukan sendiri apa yang harus dilakukan, dan bertingkah kreatif. Hal itu semua merupakan domain dari AI untuk mencoba menentukan algoritma apa yang dibutuhkan untuk memenuhi kebutuhan diatas.

Dalam bidang akademik, beberapa peniliti AI termotivasi oleh filosofi, yaitu memahami alam pikiran dan alam kecerdasan dan membangun program untuk memodelkan bagaimana proses berpikir. Beberapa juga termotivasi oleh *psychology*, bertujuan untuk memahami mekanisme otak manusia dan proses mental. Dan lainnya termotivasi oleh *engineering*, dengan tujuan membangun algoritma untuk melakukan kegiatan seperti manusia atau hewan.

Dalam pembangunan *game*, umumnya akan cenderung hanya pada sisi *engineering* yang bertujuan membangun algoritma yang dapat membuat *game* karakter mengerjakan kegiatan seperti yang dilakukan manusia atau binatang. [2]

2.2 Algoritma *Fuzzy Logic*

Fuzzy logic adalah suatu cara yang tepat untuk memetakan suatu ruang *input* ke dalam ruang *output*. Untuk sistem yang sangat rumit, penggunaan logika *fuzzy* (*fuzzy logic*) adalah salah satu pemecahannya. Sistem tradisional dirancang untuk mengontrol keluaran tunggal yang berasal dari beberapa masukan yang tidak saling berhubungan. Karena ketidak tergantungan ini, penambahan masukan yang baru akan memperumit proses kontrol dan membutuhkan proses perhitungan kembali dari semua fungsi . Kebalikannya, penambahan masukan baru pada sistem *fuzzy*, yaitu sistem yang bekerja berdasarkan prinsip-prinsip logika *fuzzy*, hanya membutuhkan penambahan fungsi keanggotaan yang baru dan aturanaturan yang berhubungan dengannya.

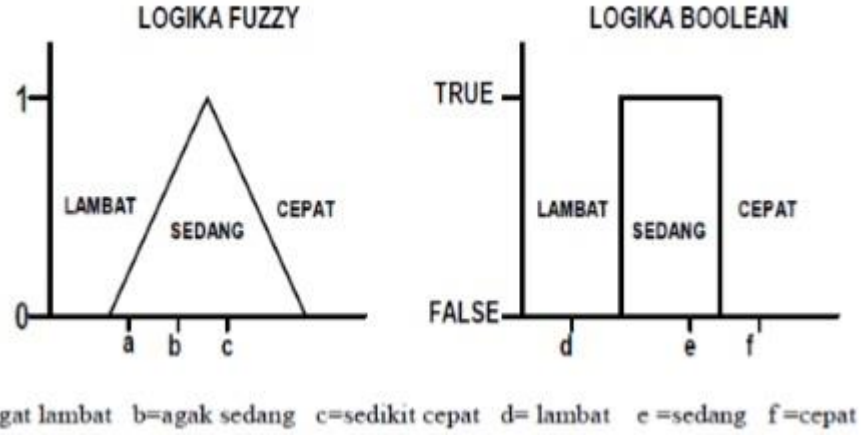
Secara umum, sistem *fuzzy* sangat cocok untuk penalaran pendekatan terutama untuk sistem yang menangani masalah-masalah yang sulit didefinisikan dengan menggunakan model matematis. Misalkan, nilai masukan dan parameter sebuah sistem bersifat kurang akurat atau kurang jelas, sehingga sulit mendefinisikan model matematikanya.

Sistem *fuzzy* mempunyai beberapa keuntungan bila dibandingkan dengan sistem tradisional, misalkan pada jumlah aturan yang dipergunakan. Pemrosesan awal sejumlah besar nilai menjadi sebuah nilai derajat keanggotaan pada sistem *fuzzy* mengurangi jumlah nilai menjadi sebuah nilai derajat keanggotaan pada sistem *fuzzy* mengurangi jumlah nilai yang harus dipergunakan pengontrol untuk membuat suatu keputusan. Keuntungan lainnya adalah sistem *fuzzy* mempunyai kemampuan penalaran yang mirip dengan kemampuan penalaran manusia. Hal ini disebabkan karena sistem *fuzzy* mempunyai kemampuan untuk memberikan respon berdasarkan informasi yang bersifat kualitatif, tidak akurat, dan ambigu.

Sistem *fuzzy* pertama kali diperkenalkan oleh Prof. L. A. Zadeh dari Barkelay pada tahun 1965. Sistem *fuzzy* merupakan penduga numerik yang terstruktur dan dinamis. Sistem ini mempunyai kemampuan untuk mengembangkan sistem intelijen dalam lingkungan yang tak pasti. Sistem ini menduga suatu fungsi dengan logika *fuzzy*. Dalam logika *fuzzy* terdapat beberapa proses yaitu penentuan himpunan *fuzzy*, penerapan aturan IF-THEN dan proses inferensi *fuzzy*. [4]

2.2.1 Himpunan *Fuzzy*

Himpunan *fuzzy* merupakan suatu pengembangan lebih lanjut tentang konsep himpunan dalam matematika. Himpunan *fuzzy* adalah rentang nilai-nilai. Masing-masing nilai mempunyai derajat keanggotaan (*membership*) antara 0 sampai dengan 1. Ungkapan logika Boolean menggambarkan nilai-nilai "benar" atau "salah". Logika *fuzzy* menggunakan ungkapan misalnya : "sangat lambat", "agak sedang", "sangat cepat"dan lain-lain untuk mengungkapkan derajat intensitasnya. Ilustrasi antara keanggotaan *fuzzy* dengan Boolean set dapat dilihat pada Gambar 2.1: [4]



Gambar 2.1 Pendefinisian Kecepatan Dalam Bentuk *Fuzzy Logic* [3]

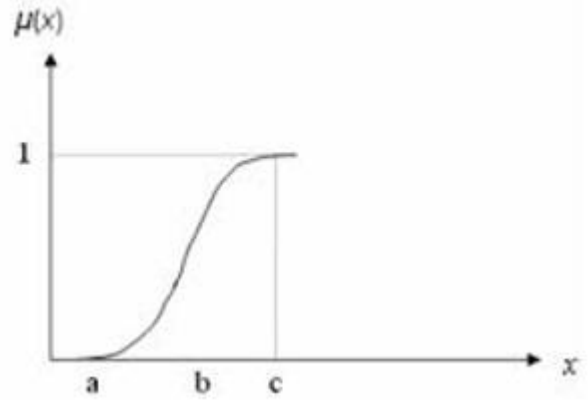
2.2.1.1 Fungsi-Fungsi Keanggotaan

Di dalam *fuzzy* sistem, fungsi keanggotaan memainkan peranan yang sangat penting untuk merepresentasikan masalah dan menghasilkan keputusan yang akurat. Terdapat banyak sekali fungsi keanggotaan yang biasa digunakan. Disini hanya membahas empat fungsi keanggotaan yang sering digunakan di dunia nyata, yaitu: [3]

1. Fungsi Sigmoid

Sesuai dengan namanya, fungsi ini berbentuk kurva sigmoidal seperti huruf S. Setiap nilai x (anggota *crisp* set) dipetakan ke dalam interval [0,1]. Grafik dan notasi matematika untuk fungsi sigmoid dapat dilihat pada Gambar 2.2.

$$Sigmoid\ (a,b,c) = \begin{cases} 0, x \leq a \\ 2\left(\frac{x-a}{c-a}\right)^2, a < x \leq b \\ 1 - 2\left(\frac{x-a}{c-a}\right)^2, b < x \leq c \\ 1, c < x \end{cases} \quad (2.1)$$

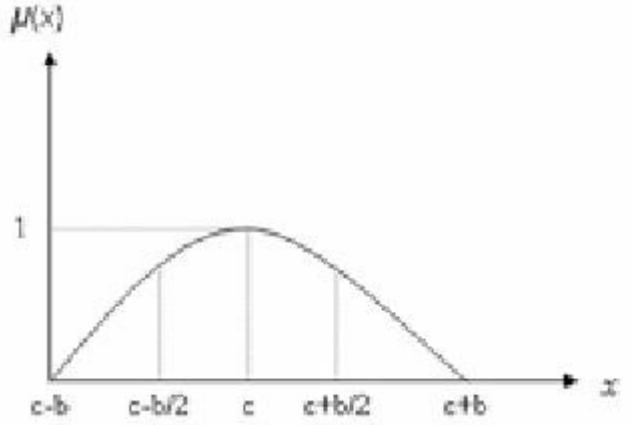


Gambar 2.2 Grafik Dan Notasi Fungsi Sigoid [3]

2. Fungsi Phi

Pada fungsi keanggotaan ini, hanya terdapat satu nilai x yang memiliki derajat keanggotaan yang sama dengan 1, yaitu ketika x=c. Nilai-nilai di sekitar c memiliki derajat keanggotaan yang masih mendekati 1. Grafik dan notasi matematika untuk fungsi phi dapat dilihat pada Gambar 2.3.

$$Phi\ (x,b,c) = \begin{cases} Phi\left(x, c-b, c-\frac{b}{2}, c\right), x \leq c \\ 1 - Phi\left(x, c, c+\frac{b}{2}, c+b\right), x > c \end{cases} \quad (2.2)$$

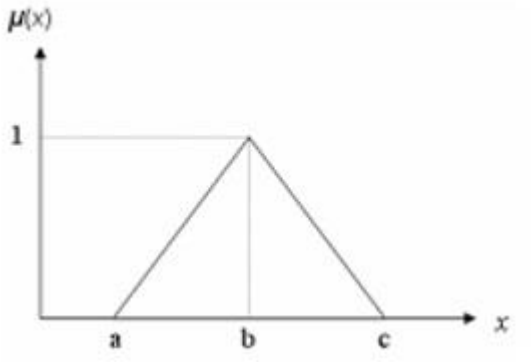


Gambar 2.3 Grafik Dan Notasi Fungsi Phi [3]

3. Fugsi Segitiga

Sama seperti fungsi phi, pada fungsi ini juga terdapat hanya satu nilai x yang memiliki derajat keanggotaan sama dengan 1, yaitu ketika x=b. Tetapi, nilai-nilai di sekitar b memiliki derajat keanggotaan yang turun cukup tajam menjauhi 1. Grafik dan notasi matematika untuk fungsi segitiga dapat dilihat pada Gambar 2.4.

$$Segitiga\ (x,a,b,c) = \begin{cases} 0, x \leq a, x \geq c \\ \dfrac{x-a}{b-a}, a < x \leq b \\ \dfrac{c-x}{c-b}, b < x \leq c \end{cases} \quad (2.3)$$

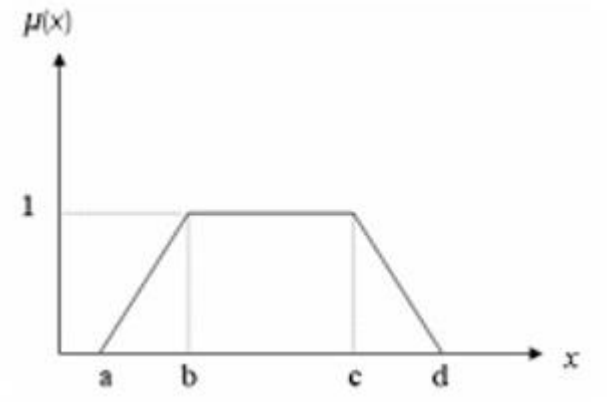


Gambar 2.4 Grafik Dan Notasi Fungsi Segitiga [3]

4. Fungsi Trapesium

Berbeda dengan fungsi segitiga, pada fungsi ini terdapat beberapa nilai x yang memiliki derajat keanggotaan sama dengan 1, yaitu ketika $b \leq x \leq c$. Tetapi derajat keanggotaan untuk $a \leq x \leq b$ dan $c \leq x \leq d$ memiliki karakteristik yang sama dengan fungsi segitiga. Grafik dan notasi matematika untuk fungsi ini dapat dilihat pada Gambar 2.5.

$$Trapesium\ (x,a,b,c,d) = \begin{cases} 0, x \leq a, x \geq d \\ \dfrac{x-a}{b-a}, a \leq x \leq b \\ 1, b \leq x \leq c \\ \dfrac{d-x}{d-c}, c \leq x \leq d \end{cases} \quad (2.4)$$



Gambar 2.5 Grafik Dan Notasi Fungsi Trapesium [3]

Himpunan fuzzy memiliki 2 atribut yaitu:

1. Linguistik, yaitu penamaan suatu grup yang mewakili suatu keadaan atau kondisi tertentu dengan menggunakan bahasa alami, seperti: MUDA, PAROBAYA, TUA
2. Numeris, yaitu suatu nilai (angka) yang menunjukkan ukuran dari suatu variabel seperti: 40, 25, 50, dsb.

2.2.2 Variabel Linguistik

Variabel linguistik adalah suatu interval numerik dan mempunyai nilai- nilai linguistic, yang semantiknya di definisikan oleh fungsi keanggotaannya. Misalnya, suhu adalah suatu variabel linguistik yang bisa di definisikan pada interval [-10°C, 40°C]. Variabel tersebut bisa memiliki nilai-nilai linguistik seperti ‘Dingin’, ‘Hangat’, ‘Panas’ yang semantiknya di definisikan oleh fungsi-fungsi keanggotaan tertentu. [3]

Suatu sistem berbasis aturan *fuzzy* yang lengkap terdiri dari 3 komponen utama dapat dilihat pada Gambar 2.6: [3]

1. *Fuzzyfication*

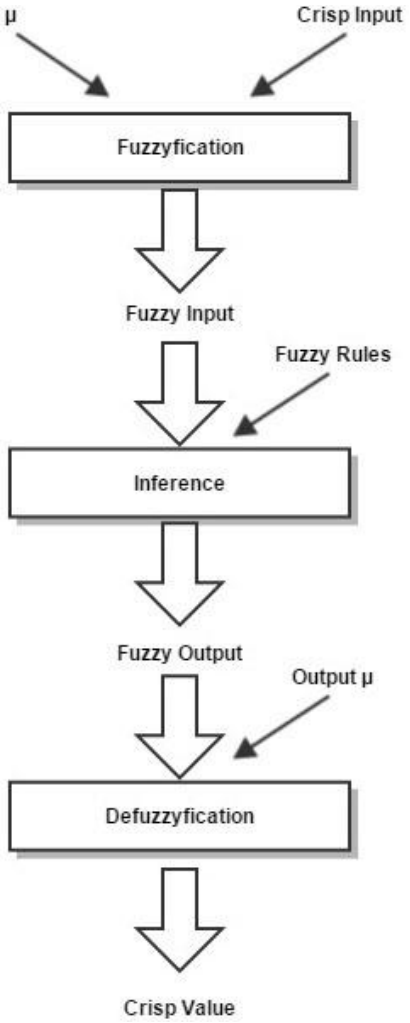
   Mengubah masukan-masukan yang nilai kebenarannya bersifat pasti (*crips input*) kedalam bentuk *fuzzy input*, yang berupa nilai linguistik yang semantiknya ditentukan berdasarkan fungsi keanggotaan tertentu.

2. *Inference*

   Melakukan penalaran menggunakan *fuzzy input* dan *fuzzy rules* yang telah di tentukan sehingga menghasilkan *fuzzy output*.

3. *Defuzzyfication*

   Mengubah *fuzzy output* menjadi *crisp value* berdasarkan fungsi keanggotaan yang telah ditentukan.

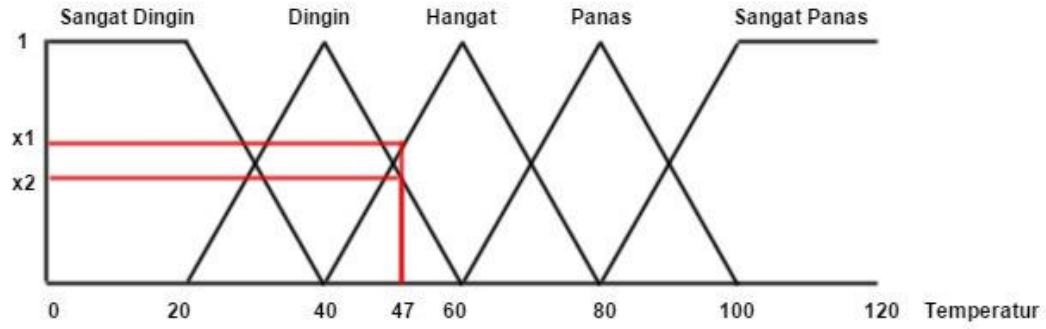


Gambar 2.6 Diagram Blok Sistem Berbasis Aturan *Fuzzy* [3]

2.2.3 Fuzzyfikasi

Fuzzifikasi yaitu suatu proses untuk mengubah suatu masukan dari bentuk tegas (*crisp*) menjadi *fuzzy* (variabel linguistik) yang biasanya disajikan dalam bentuk himpunan-himpunan *fuzzy* dengan suatu fungsi kenggotaannya masing-

masing. Contoh dari proses *fuzzification* adalah seperti yang ditunjukkan di gambar 2.7. Sebuah sistem *fuzzy* untuk mengukur suhu mempunyai 5 buah *membership function* yang mempunyai label sangat dingin, dingin, hangat, panas, sangat panas. Kemudian input yang diperoleh dari *crisp* input adalah 47° maka pengambilan *fuzzy input*-nya adalah seperti pada Gambar 2.7



Gambar 2.7 Proses Perubahan Dari Crisp Input Menjadi Fuzzy Input

Sehingga didapat 2 *fuzzy* input yang masing-masing adalah: dingin (x_2) dan hangat (x_1). Nilai x_1 dan x_2 dapat dicari dengan rumus persamaan garis. Yang menentukan sistem anda sensitif atau tidak adalah *membership function* ini. Jika *membership function*-nya banyak maka sistem anda menjadi sensitif. Yang dimaksud dengan sensitif dalam hal ini adalah jika input-nya berubah sedikit saja maka sistem akan cepat merespon dan menghasilkan suatu *output* lain. *Output* dari proses *fuzzyfication* ini adalah sebuah nilai *input fuzzy* atau yang biasanya dinamakan *fuzzy input.*

2.2.4 *Inference*

Dalam suatu sistem aturan *fuzzy*, proses *inference* memperhitungkan semua aturan yang ada dalam basis pengetahuan. Hasil dari proses *inference* direpresentasikan oleh suatu *fuzzy* set untuk setiap variabel bebas (pada *consequent*). Derajat keanggotaan untuk setiap nilai variabel tidak bebas menyatakan ukuran kompatibilitas terhadap variabel bebas (pada *antecdent*). Misalkan, terdapat suatu sistem dengan *n* variabel x_1, ..., x_n dan *m* variabel tidak bebas $y_1,...,y_m$. Misalkan *R* adalah suatu basis dari sejumlah *r* aturan *fuzzy*.

IF $P_1(x_1,...,x_n)$ THEN Q_1 $(y_1,...,y_m)$,

IF $P_r(x_1,...,x_n)$ THEN Q_r $(y_1,...,y_m)$,

Dimana $P_1,...P_r$ menyatakan *fuzzy predicate* untuk variabel bebas, dan $Q_1,...Q_r$ menyatakan *fuzzy predicate* untuk variabel tidak bebas. [5]

Struktur sistem inferensi *fuzzy* dapat dilihat pada gambar 2.8



Gambar 2.8 Struktur Sistem Inferensi *Fuzzy* [2]

Keterangan:

1) Basis Pengetahuan *Fuzzy* merupakan kumpulan *rule-rule fuzzy* dalam bentuk pernyataan IF…THEN.
2) Fuzzyfikasi adalah proses untuk mengubah input sistem yang mempunyai nilai tegas menjadi variabel *linguistic* menggunakan fungsi keanggotaan yang disimpan dalam basis pengetahuan *fuzzy*.
3) Logika pengambil keputusan merupakan proses untuk mengubah input *fuzzy* dengan cara mengikuti aturan-aturan (IF-THEN *Rules*) yang telah ditetapkan pada basis pengetahuan *fuzzy*.
4) Defuzzyfikasi merupakan proses mengubah output *fuzzy* yang diperoleh dari mesin inferensi menjadi nilai tegas menggunakan fungsi keanggotaan yang sesuai dengan saat dilakukan fuzzyfikasi.

Terdapat dua model aturan *fuzzy* yang digunakan secara luas dalam berbagai aplikasi. [2]

1. Model Mamdani

   Pada model ini, aturan *fuzzy* didefinisikan sebagai:

   IF x_1 is A_1 AND …AND x_n is A_n THEN y is B

   di mana A_1, …, A_n, dan B adalah nilai-nilai linguistik (atau *fuzzy* set) dan "x_1 is A_1" menyatakan bahwa nilai x1 adalah anggota *fuzzy* set A1.

2. Model Sugeno

   Model ini dikenal juga sebagai Takagi-Sugeno-Kang (TSK) model, yaitu suatu varian dari Model Mamdani. Model ini menggunakan aturan yang berbentuk:

   IF x_1 is A_1 AND…AND x_n is A_n THEN $y=f(x_1,\ldots,x_n)$

   di mana f bisa sembarang fungsi dari variabel-variabel input yang nilainya berada dalam interval variabel output. Biasanya, fungsi ini dibatasi dengan menyatakan f sebagai kombinasi linier dari variabel-variabel input:

   $f(x_1,\ldots,x_n) = w_0 + w_1x_1 + \ldots+w_nx_n$

   di mana w_0, $w_1,\ldots,w_n$ adalah konstanta yang berupa bilangan real yang merupakan bagian dari spesifikasi aturan *fuzzy*.

2.2.5 Defuzzyfikasi

Defuzzification mengubah *fuzzy output* menjadi *crisp value* berdasarkan fungsi keanggotaan yang telah ditentukan.Terdapat berbagai metode *defuzzification* yang telah berhasil diaplikasikan untuk berbagai macam masalah, di sini dibahas 5 metode di antaranya, yaitu: [2]

1. *Centroid Method*

   Metode ini disebut juga sebagai *Center of Area* atau *Center of Gravity*. Metode ini menghitung nilai *crisp* menggunakan rumus:

   $$y* = \frac{\int y\mu_R(y)dy}{\int y\mu_R(y)dy} \quad (2.5)$$

   di mana y* suatu nilai *crisp*.

   Fungsi *integration* dapat diganti dengan fungsi *summation* jika y bernilai diskrit, sehingga menjadi:

   $$y* = \frac{\int y\mu_R(y)}{\int y\mu_R(y)} \quad (2.6)$$

   di mana y* adalah nilai *crisp* dan $\mu_R(y)$ adalah derajat keanggotaan dari y.

2. *Height Method*

   Metode ini dikenal sebagai prinsip keanggotaan maksimum karena metode ini secara sederhana memilih nilai *crisp* yang memiliki derajat keanggotaan maksimum. Oleh karena itu, metode ini hanya bisa dipakai untuk fungsi

keanggotaan yang memiliki derajat keanggotaan 1 pada suatu nilai *crisp* tunggal dan dan 0 pada semua nilai crisp yang lain. Fungsi seperti ini sering disebut sebagai *singleton*.

3. *First (or Last) of Maxima*

   Metode ini juga merupakan generalisasi dari height method untuk kasus di mana fungsi keanggotaan *output* memiliki lebih dari satu nilai maksimum. Sehingga nilai *crisp* yang digunakan adalah salah satu dari nilai yang dihasilkan dari maksimum pertama atau maksimum terakhir (tergantung pada aplikasi yang akan dibangun).

4. *Mean-Max Method*

   Metode ini disebut juga sebagai *Middle of Maxima*. Merupakan generalisasi dari *height method* untuk kasus di mana terdapat lebih dari satu nilai *crisp* yang memiliki derajat keanggotaan maksimum. Sehingga y* didefinisikan sebagai titik tengah antara nilai *crisp* terkecil dan nilai *crisp* terbesar

   $$y* = \frac{m + M}{2} \quad (2.7)$$

   di mana m adalah nilai *crisp* yang paling kecil dan M adalah nilai *crisp* yang paling besar.

5. *Weighted Average*

   Metode ini mengambil nilai rata-rata dengan menggunakan pembobotan berupa derajat keanggotaan. Sehingga y* didefinisikan sebagai:

   $$y* = \sum \frac{\mu(y)y}{\mu(y)} \quad (2.8)$$

   di mana y adalah nilai *crisp* dan $\mu(y)$ adalah derajat keanggotan dari nilai *crisp* y.

2.3 *Game*

Game merupakan kata dalam bahasa Inggris yang berarti permainan. Permainan adalah sesuatu yang dapat dimainkan dengan aturan tertentu sehingga ada yang menang dan ada yang kalah, biasanya dalam konteks tidak ada serius atau dengan tujuan *refreshing*. Suatu cara belajar yang digunakan dalam menganalisa

interaksi antara sejumlah pemain maupun perorangan yang menunjukan strategi-strategi yang rasional.

Teori permainan pertama kali ditemukan oleh sekelompok ahli Matematika pada tahun 1944. Teori itu dikemukakan oleh Jhon Von Neumann dan Oskar Morgenstern yang berisi: "Permainan terdiri atas sekumpulan peraturan yang membangun situasi bersaing dari dua sampai beberapa orang atau kelompok dengan memilih strategi yang dibangun untuk memaksimalkan kemenangan sendiri atau pun untuk meminimalkan kemenangan lawan. Peraturan-peraturan memnentukan kemungkinan tindakan untuk setiap pemain, sejumlah keterangan diterima setiap pemain sebagai kemajuan bermain, dan sejumlah kemenangan atau kekalahan dalam berbagai situasi." [6]

2.3.1 Pengertian *Game*

Pengertian *game* menurut para ahli: [7]

1. Menurut Agustinus Nilwan dalam bukunya Pemograman Animasi dan *Game Profesional* terbitan Media Komputindo, *game* merupakan permainan komputer yang dibuat dengan teknik dan metode animasi. Jika ingin mendalami penggunaan animasi haruslah memahami pembuatan *game*. Atau jika ingin membuat *game*, maka haruslah memahami teknik dan metode animasi, sebab keduanya saling berkaitan.
2. Menurut Clark C. Abt, *game* adalah kegiatan yang melibatkan keputusan pemain, berupaya mencapai tujuan dengan dibatasi oleh konteks tertentu (misalnya, dibatasi oleh peraturan).
3. Menurut Chris Crawford, seorang *computer game designer* mengemukakan bahwa *game*, pada intinya adalah sebuah interaktif. Aktivitas yang berpusat pada sebuah pencapaian, ada pelaku aktif (*player*), ada pelaku pasif (NPC).
4. Menurut David Parlett, *game* adalah sesuatu yang memiliki "akhir dan cara pencapaiannya": artinya ada tujuan, hasil dan serangkaian peraturan untuk mencapai keduanya.
5. Menurut Roger Caillois, seorang sesiolog Prancis dalam bukunya yang berjudul Les Jeux Et Les Hommes menyatakan *game* adalah aktivitas yang

mencakup karakteristik berikut: *fun* (bebas bermain adalah pilihan bukan kewajiban), *separate* (terpisah), *uncertain*, *non-productive*, *governed by rules* (ada aturan), *fictitious* (pura-pura).

6. Menurut Greg Costikyan, *game* adalah "Sebentuk karya seni dimana peserta, yang disebut pemain, membuat keputusan untuk mengelola sumber daya yang dimilikinya melalui benda di dalam *game* demi mencapai tujuan".

2.3.2 Klasifiasi *Game*

Game biasanya digunakan untuk mengkatagorikan sebuah *game* berdasarkan dari interaksi *gameplay* daripada perbedaan tampilan atau narasi. Sebuah jenis *game* dapat didefinisikan dengan kumpulan dari sebuah tipe permainan dari *game* tersebut. *Game* dapat diklasifikasikan menjadi dua, yaitu:

1. Berdasarkan Jenis *Platform* yang Digunakan

   Tiga kategori utama *platform* pada *game* adalah *personal computer* (PC), *console*, dan *mobile*:

   1. *Personal Computer* (PC)

      PC *game* adalah *game* yang dibuat untuk komputer baik berbasis Windows, Mac, ataupun Linux. PC menyediakan kekuatan grafis dan pemrosesan yang kuat yang memungkinkan pengembangan untuk membuat *game* yang mutakhir. Tetapi kelemahan PC adalah mahal karena pengguna harus menggunakan *hardware* yang *up-to date* untuk memainkan *game* PC dengan haik. Selain itu, kelemahan *game* PC bagi pengembang adalah banyaknya varian dari konfigurasi PC membuat sulit bagi pengembangan untuk memastikan *game* tersebut berjalan dengan benar pada semua *setup* PC.

   2. *Console*

      Console adalah *hardware* yang dibuat oleh pihak ketiga seperti Sony, Microsoft, dan Nintendo. *Console* terhubung ke televisi dan tujuan utamanya adalah untuk bermain *game*. *Game Console* sangat menarik bagi *game* pengembang karena mereka hanya perlu memikirkan satu

konfigurasi *hardware* ketika membuat *software* untuk konsol. Sangat kontras dengan PC yang memiliki opsi konfigurasi yang tak terbatas.

3. *Mobile*

   Mobile platform terdiri atas sesuatu yang *portable* dan bisa digenggam, termasuk ponsel, PDA, ipod, dan *handle game* seperti Nintendo Dsi atau Sony PSP. *Game mobile* memiliki kontrol yang sederhana (terutama jika dibandingkan dengan PC).

2. Berdasarkan *Genre* Permainan

   Genre game bisa dibagi menjadi 9, yaitu: [8]

   1. Action

      Action game adalah *game* dimana kebanyakan dari tantangan yang disajikan merupakan dari tes *physical skill* dan koordinasi pemain. Salah satu *sub-genre action game* adalah *shooters game*, baik yang 2D maupun 3D seperti *First Person, Shooters* (FPS).

   2. Strategy

      Strategy game menantang pemain untuk mencapai kemenangan dengan perencanaan, khususnya melalui perencanaan serangkaian tindakan yang dilakukan melawan satu lawan atau lebih. Kemenangan diraih dengan perencanaan matang dan pengambilan keputusan yang optimal.

   3. *Role Playing Game* (RPG)

      RPG adalah *game* dimana pemain mengontrol satu atau lebih karakter yang biasanya di desain oleh pemain itu sendiri, dan memandu mereka melewati berbagai rintangan yang diatur oleh komputer. Perkembangan karakter dalam hal kekuatan dan kemampuannya adalah kunci dari *game* jenis ini.

   4. Sports

      Sports game mensimulasikan berbagai aspek dari olahraga atletik nyata maupun imajiner, apakah itu memainkan pertandingan, *memanage* tim dan karir atau keduanya. Salah satu contoh *game* jenis ini adalah *Pro Evolution Soccer* 2012 (PES 2012), dimana pemain bisa memainkan pertandingan,

menjadi manajer tim, maupun menjadi pemain dan mengembangkan karirnya sendiri.

5. *Vehicle Simulation*

   Vechicle Simulation membuat *feeling* mengedarai kendaraan, baik kendaraan nyata maupun kendaraan imajiner. Performa dan karakteristik kasar mesin harus menyerupai kenyataan, kecuali jika yang didesain adalah kendaraan imajiner.

6. *Construction and Management Simulation*

   CMS *game* adalah *game* tentang proses. Tujuan pemain bukan untuk mengalahkan musuh, tetapi membangun sesuatu dengan konteks proses yang sedang berjalan. Semakin pemain mengerti dan mengontrol proses, semakin sukses sesuatu yang ia bangun. *Game* seperti ini biasanya menyediakan dua jenis permainan, yaitu mode bebas dimana pemain bebas membangun sesuatu dan mode misi dimana terdapat skenario hal apa yang harus dibangun oleh pemain.

7. *Adventure*

   Adventure game adalah cerita interaktif tentang karakter *protagonist* yang dimainkan oleh pemain. Penyampaian cerita dan eksplorasi adalah elemen inti dari *game* ini. Penyelesaian tekateki dan tantangan konseptual adalah bagian besar dari permainan.

8. *Artificial Life and Puzzle Game*

   Artificial Life game adalah *game* yang membuat tiruan dari kehidupan sebenarnya. Biasanya ada dua jenis *game* ini, tiruan kehidupan manusia, contohnya *The SIMS*, dan tiruan kehidupan binatang, contohnya *Tamagochi*.

9. *Online Game*

   Istilah online *game* disini mengacu kepada *multiplayer game* dimana mesin dari para pemain terhubung dengan jaringan.

3. Kategori *Game* lainnya

   Selain berdasarkan *genre* permainannya terdapat pula kategori-kategori *game* lainnya, yaitu: [8]

1. *Multiplayer Online*

   Game yang dapat dimainkan secara bersamaan oleh lebih dari 2 orang (bahkan dapat mencapai puluhan ribu orang dalam satu waktu) membuat pemain dapat bermain bersama dalam satu dunia virtual dari sekedar *chatting* hingga membunuh naga bersama teman yang entah bermain di mana. Umumnya permainan tipe ini dimainkan di PC dan bertema RPG, walau ada juga yang bertema musik atau action.

2. Casual *Game*s

   Sesuai namanya, *game* casual itu tidak kompleks, mainnya rileks dan sangat mudah untuk dipelajari (bahkan cenderung langsung bisa dimainkan). Jenis ini biasanya memerlukan spesifikasi komputer yang standar. *Genre* permainannya biasanya *puzzle* atau *action* sederhana dan umumnya dapat dimainkan hanya menggunakan *mouse* (biasanya *game* lain menggunakan banyak tombol tergantung *game*-nya). Contohnya: *Dinner Dash, Zuma, Feeding Frenzy*. Fitur dasar pada *game casual*, yaitu:

   1) *Gameplay* sangat sederhana, bisa dimainkan menggunakan *mouse, keyboard* atau *keypad* ponsel.
   2) Dimainkan dalam waktu singkat. Biasanya dibawah 10 menit. Bisa dimainkan saat istirahat kerja atau pada transportasi umum.
   3) Kemampuan yang cepat dalam mencapai tahap akhir tanpa perlu *save game*.
   4) Kadang digunakan untuk model bisnis atau periklanan.
   5) Cocok dalam *genre* apapun.
   6) Dapat dimainkan oleh siapa saja, anak-anak sampai orang dewasa.
   7) Dapat dimainkan dengan atau tanpa pengalaman dalam bermain *game*.

3. *Edugame*s

   Video game jenis ini dibuat dengan tujuan spesifik sebagai alat pendidikan, entah untuk belajar mengenal warna untuk balita, mengenal huruf dan angka, matematika, sampai belajar bahasa asing. *Developer* yang membuatnya, harus memperhitungkan berbagai hal agar *game* ini benar-

benar dapat mendidik, menambah pengetahuan dan meningkatkan ketrampilan yang memainkannya. Target segmentasi pemain harus pula disesuaikan dengan tingkat kesulitan dan desain visual ataupun animasinya.

4. *Advergame*s

   Jenis *game* yang biasanya mudah dimainkan ini mengusung dan menampilkan produk atau *brand* secara gamblang maupun tersembunyi. Di era tumbuhnya media-media baru berteknologi tinggi sekarang ini, dunia periklanan memang sudah tidak lagi terbatas pada TV, koran, majalah, *billboard* dan radio, *video game* sekarang telah menjadi sarana beriklan atau membangun *brand awareness* yang efektif. Baik melalui internet maupun di mainkan di *event-event* mereka, *edugame*s terasa semakin dibutuhkan untuk menjaring calon konsumen bagi produk yang menggunakan *advergame*s ini.

2.3.3 Sudut Pandang *Game*

Sebuah *game* biasanya mempunyai sudut pandang permainan tersendiri disesuaikan berdasarkan *genre game* yang diambil. Berikut beberapa macam sudut pandang permainan yang biasa digunakan: [8]

1. *Side Scrolling*

   Adalah sudut pandang permainan yang terlihat dari samping dan memungkinkan karakter utama untuk bergerak dari kiri ke kanan serta memungkinkan *background* pada *game* seolah-olah bergeser mengikuti pergerakan karakter utama.

2. *Top Down*

   Adalah sudut pandang permainan yang memungkinkan karakter utama bermanuver ke empat arah namun cara permainannya sendiri bergeser dari bawah ke atas, dan biasanya *game* yang menggunakan sudut pandang permainan jenis ini adalah *shooter game*.

3. *Isometric*

   Adalah sudut pandang permainan yang memungkinkan permainan terlihat diantara sisi *Side Scrolling* dan juga *Top Down*, dan biasanya diterapkan pada *game* dengan *genre* RTS (*Real Time Strategy*).

2.3.4 Elemen Pada *Game*

Elemen-elemen yang ada dalam sebuah *game* biasanya terdiri dari: [8]

1. Desain *game*

   Desain adalah langkah awal untuk membuat semua elemen *game*. Desain *game* dibuat semenarik mungkin agar pemain tidak cepat bosan yang dampaknya membuat *game* tersebut cepat ditinggalkan.

2. Pemrograman *game*

   Pemrograman *game* sebagian besar digunakan untuk mengontrol gerakan objek di layar. Selain itu, pemrograman *game* juga digunakan untuk pemrograman suara, *input device*, deteksi keadaan lain - lain.

3. Grafis *game*

   Grafis *game* memegang peranan penting dalam pembuatan tampilan. Tampilan haruslah dibuat semenarik mungkin, sehingga dengan melihatnya saja *end user* langsung tertarik untuk memainkannya.

4. Musik dan *sound*

   Musik dan *sound* dalam suatu *game* merupakan suatu hal yang wajib dikarenakan dalam suatu *game* akan lebih terasa efek emosi dari *game* tersebut, sehingga akan membuat kesan *game* tersebut lebih terasa nyata.

2.4 OOP (*Object Oriented Programming*)

Secara spesifik, pengertian berorientasi objek berarti bahwa mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya. Hal ini yang membedakan dengan pemograman konvensional dimana struktur data dan perilaku hanya berhubungan secara terpisah. Terdapat beberapa cara untuk menentukan karakteristik dalam pendekatan berorientasi

objek, tetapi secara umum mencakup empat hal, yaitu identifikasi, klasifikasi, *polymorphism* (polimorfisme) dan *inheritance* (pewarisan). [9]

2.4.1 Konsep Dasar Berorientasi Objek

Pendekatan berorientasi objek merupakan suatu teknik atau cara pendekatan dalam melihat permasalahan dan sistem (sistem perangkat lunak, sistem informasi atau sistem lainnya). Pendekatan berorientasi objek akan memandang sistem yang akan dikembangkan sebagai suatu kumpulan objek yang berkorespodensi dengan objek-objek dunia nyata.

Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodologi berorientasi objek: [10]

1. Kelas (*Class*)

   Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statis dan himpunan objek yang sama yang mungkin lahir atau tercipta dan kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (*relationship*) dan arti. Suatu kelas dapat diwariskan ke kelas yang baru.

2. Objek (*Object*)

   Objek adalah abstraksi dan sesuatu yang mewakilkan dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, status atau hal-hal lain yang bersifat abstrak. Objek merupakan entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakukan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yang diciptakan, dimanipulasi, dan dihancurkan.

3. Metode (*Method*)

   Operasi atau metode pada sebuah kelas hampir sama dengan fungsi prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi yang befungsi untuk memanipulasi objek itu sendiri. Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek.

4. Atribut (*Attribute*)

   Atribut dari sebuah kelas adalah variable global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama dan sebagainya.

5. Abstraksi (*Absraction*)

   Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

6. Enkapsulasi (*Encapsulation*)

   Pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.

7. Pewarisan (*Inheritance*)

   Mekanisme yang memungkinkan suatu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dirinya.

8. Antarmuka (*Interface*)

   Antarmuka (*interface*) sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah interface dapat diimplementasikan oleh kelas lain.

9. *Reusability*

   Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

10. Generalisasi dan Spesialisasi

    Menunjukan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus. Misalnya kelas yang lebih umum (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil, motor, dan kereta.

11. Komunikasi Antar Objek

    Komunikasi Antar Objek dilakukan lewat pesan (*message*) yang dikirim dari suatu objek ke objek yang lain.

12. Polimorfisme (*Polymorphism*)

    Kemampuan suatu objek untuk digunakan di banyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

13. *Package*

    Package adalah sebuah container atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam package yang berbeda.

2.4.2 Analisis Berorientasi Objek

Analisis berorientasi objek atau *Object Oriented Analysis* (OOA) adalah tahap untuk menganalisis spesifikasi atau kebutuhan akan sistem yang akan dibangun dengan konsep berorientasi objek, apakah benar kebutuhan yang ada dapat diimplementasikan menjadi sebuah sistem berorientasi objek.

OOA biasanya menggunakan kartu CRC (*Component, Responsibility, Collaborator*) untuk membangun kelas-kelas yang akan digunakan atau menggunakan UML (*Unifed Modeling Language*) pada bagian diagram *use case*, diagram kelas, dan diagram objek. [10]

2.4.3 Desain Berorientasi Objek

Desain berorientasi objek atau *Object Oriented Design* (OOD) adalah tahapan perantara untuk spesifikasi atau kebutuhan sistem yang akan dibangun dengan konsep berorientasi objek ke desain pemodelan agar lebih mudah diimplementasikan dengan pemograman berorientasikan objek.

Pemodelan berorientasi objek biasanya dituangkan dalam dokumentasi perangkat lunak dengan menggunakan perangkat lunak dengan menggunakan perangkat pemodelan berorientasi objek, diantaranya adalah UML (*Unified Modeling Language*). Kendala dan permasalahan pembangunan sistem berorientasi objek biasanya dapat dikenali dalam tahap ini. [10]

2.4.4 Pemodelan

Pemodelan adalah gambaran dari realita yang sederhana dan dituangkan dalam bentuk pemetaan dengan aturan tertentu. Pemodelan dapat menggunakan bentuk yang sama dengan realitas. Pemodelan juga banyak digunakan untuk merencanakan suatu hal agar kegagalan dan resiko yang mungkin terjadi dapat meminimalisir.

Pada dunia pembangunan perangkat lunak juga diperlunakan pemodelan. Pemodelan perangkat lunak digunakan untuk mempermudah langkah berikutnya dari pengembangan sebuah sistem sehingga lebih terencana. Pemodelan pada pembangunan perangkat lunak untuk menvisualkan perangkat lunak yang akan dibuat. [10]

2.5 UML (*Unified Modelling Language*)

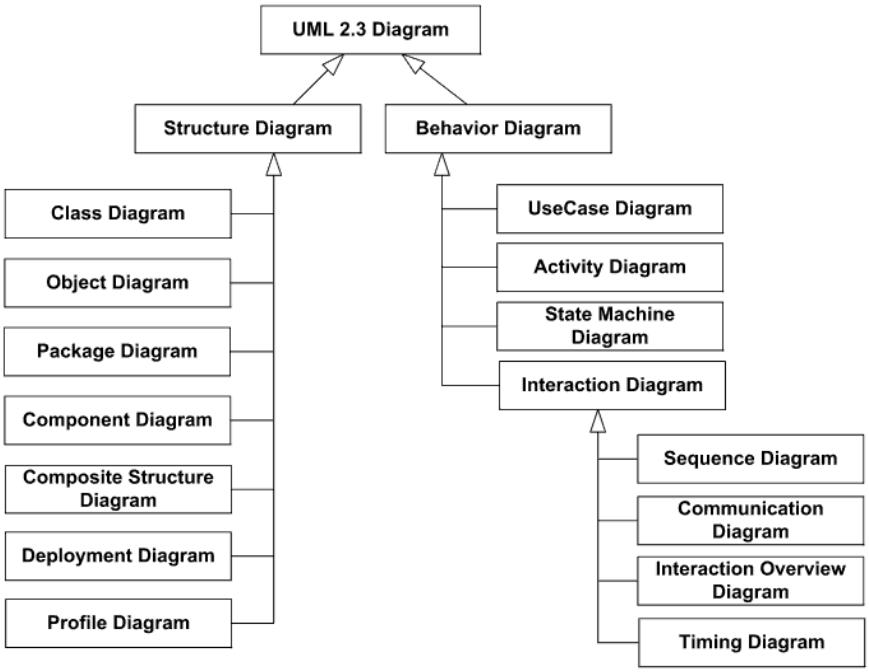
UML (*Unified Modelling Language*) adalah sebuah "bahasa" yang telah menjadi standar dalam industri visualisasi, merancang, dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

UML dapat digunakan untuk membuat model untuk semua jenis perangkat lunak, dimana perangkat lunak tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa berorientasi objek.

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentum, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya yaitu : Grady Booch OOD (*Object Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*) dan Ivar Jacobson OOSE (*Object Oriented Software Engineering*). [10]

2.5.1 Diagram UML

Pada UML 2.3 terdiri dari 13 macam diagram yang dikelompokan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar 2.9.



Gambar 2.9 Diagram UML [10]

Berikut ini penjelasan singkat dari pembagian kategori tersebut. [10]

1. *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
3. *Interaction* diagrams yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antarsubsistem pada suatu sistem.

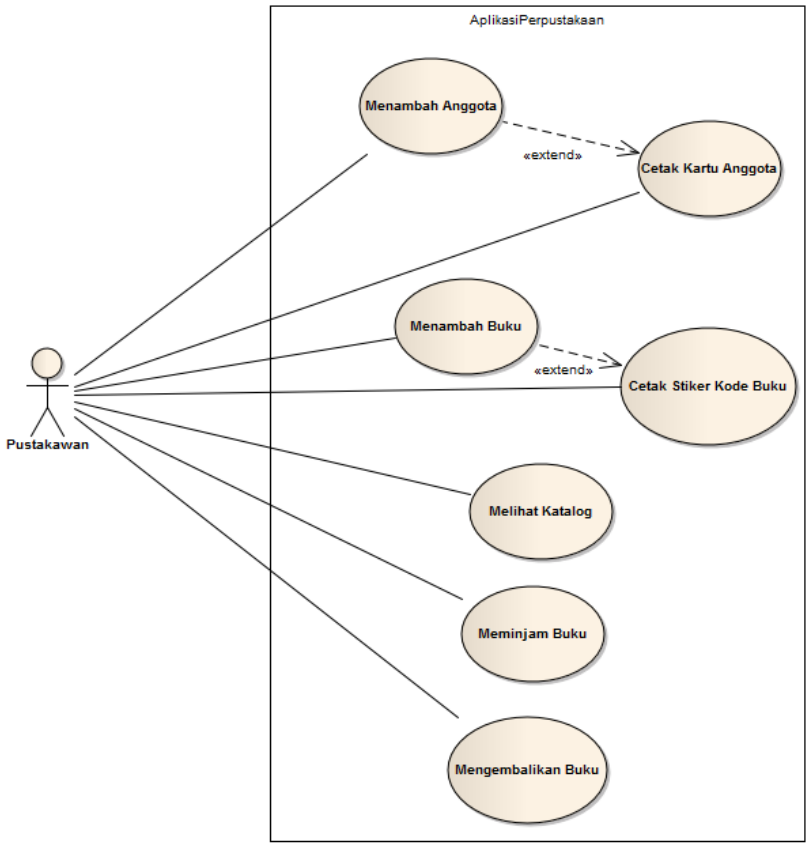
2.5.2 Diagram *Use Case* (*Use Case Diagram*)

Diagram *use case* merupakan pemodelan untuk kelakuan (*behaviour*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case*

digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*. [10]

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unitunit yang saling bertukar pesan antar unit atau aktor.

Contoh *Use Case* Diagram dapat dilihat pada Gambar 2.10



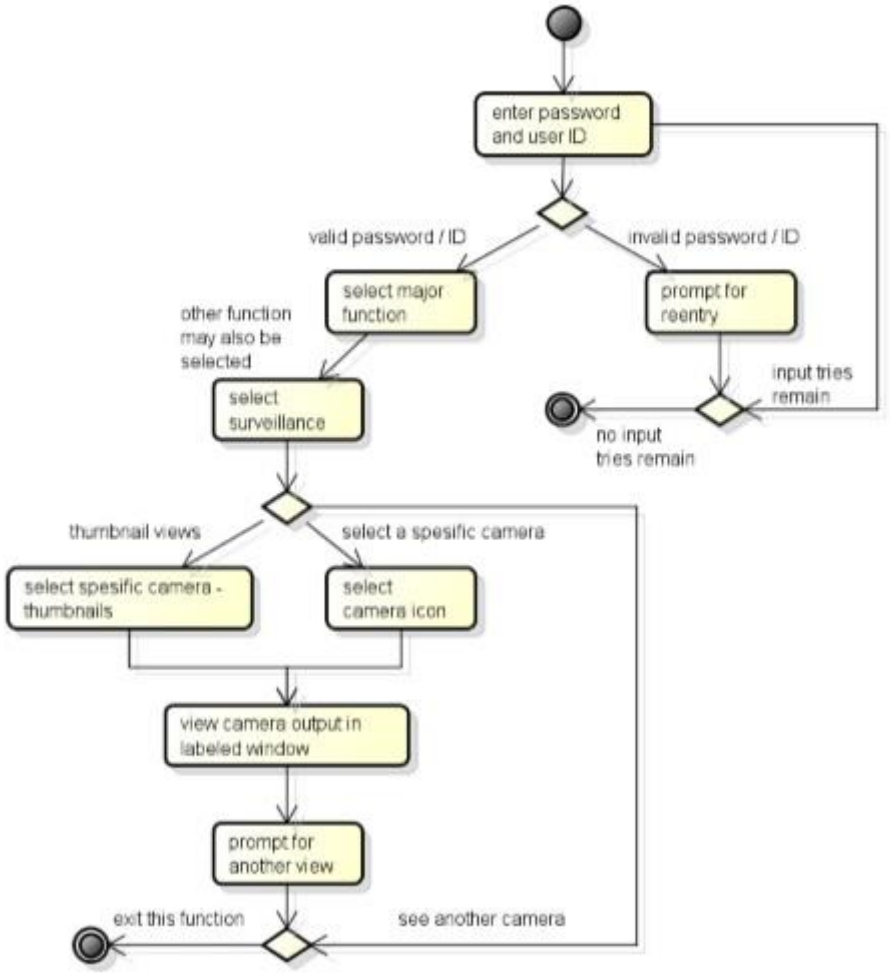
Gambar 2.10 Contoh *Use Case Diagram*

2.5.3 Diagram Aktivitas (*Activity Diagram*)

Diagram aktivitas menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut: [10]

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokkan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4. Rancangan menu yang ditampilkan pada perangkat lunak.

Contoh *Activity Diagram* dapat dilihat pada Gambar 2.11

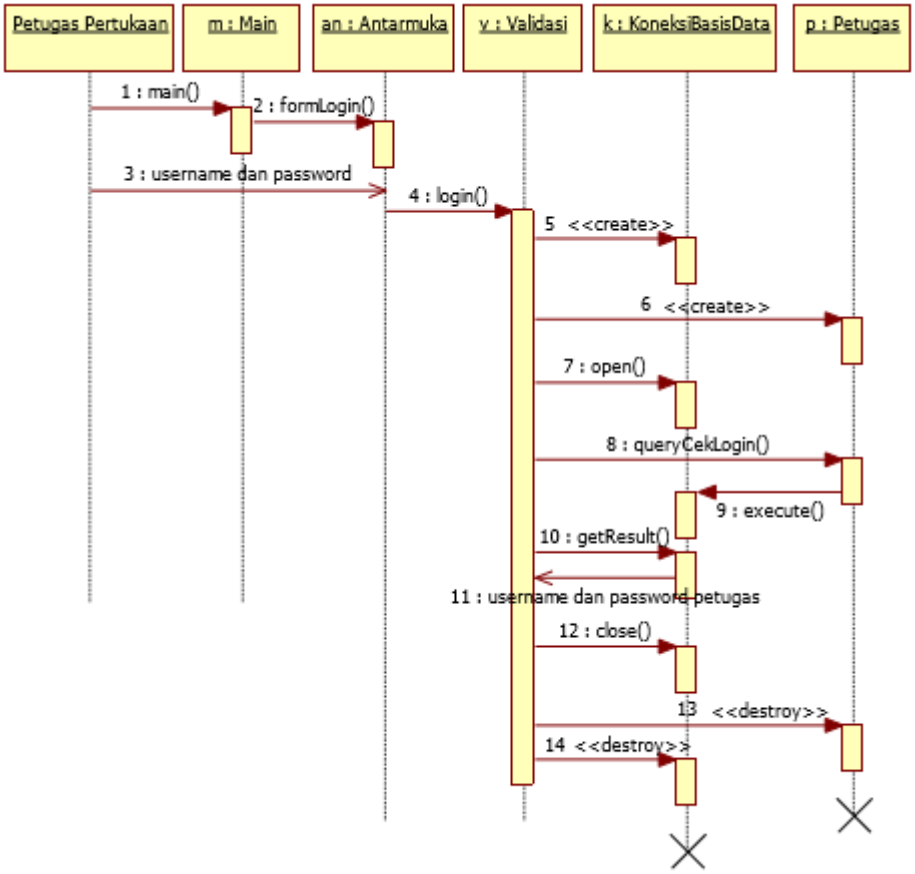


Gambar 2.11 Contoh *Activity Diagram*

2.5.4 Diagram Sekuen (*Sequence Diagram*)

Diagram sekuen menggambarkan kelakuan pada objek *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case*. Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua use case yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus

dibuat juga semakin banyak. Contoh *Sequence Diagram* dapat dilihat pada Gambar 2.12. [10]



Gambar 2.12 Contoh *Sequences Diagram*

2.5.5 Diagram Kelas (*Class Diagram*)

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas mimiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas. Contoh *Class Diagram* dapat dilihat pada Gambar 2.13. [10]



Gambar 2.13 Contoh *Class Diagram*

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*.

a. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem yang akan dibuat diluar sistem yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
b. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor. [10]

2.6 *Unity*

Unity Game Engine adalah *software* atau *game engine* yang digunakan untuk membuat *video game* berbasis dua atau tiga dimensi dan dapat digunakan secara gratis. Selain untuk membuat *game*, *unity* juga dapat digunakan untuk membuat konten yang interaktif lainnya seperti, visual arsitektur dan *real-time* 3D animasi.

Unity adalah sebuah sebuah *tool* yang terintegrasi untuk membuat *game*, arsitektur bangunan dan simulasi. *Unity* bisa digunakan untuk *games* PC dan *games online*. Untuk *games online* diperlukan sebuah *plugin*, yaitu *Unity Web Player*, yang sama halnya dengan *flash player* pada *browser*. Bahasa pemrograman yang digunakan bermacam-macam, mulai dari *JavaScript*, C#, dan Boo.

Unity tidak bisa melakukan desain atau *modelling*, dikarenakan *unity* bukan merupakan *tools* untuk mendesain. Banyak hal yang bisa di lakukan di *unity*, ada fitur *audio reverb zone*, *particle effect*, *sky box* untuk menambahkan langit, dan

masih banyak lagi, dan juga bisa langsung *edit texture* dari *editor* seperti *photoshop* dll.

Features (*Scripting*) di dalam *unity* adalah sebagai berikut (*Unity Technologies*, 2013):

1. Mendukung 3 bahasa pemrograman, *JavaScript*, C#, dan *Boo*.
2. *Flexible* and *EasyMoving*, *rotating*, dan *scaling objects* hanya perlu sebaris kode. Begitu juga dengan *duplicating*, *removing*, dan *changing properties*.
3. *Multi Platform Game* bisa di *deploy* di PC, Mac, Wii, iPhone, iPad dan *browser*, android.
4. *Visual Properties Variables* yang di definisikan dengan *scripts* ditampilkan pada *editor*. Bisa digeser, di *drag* and *drop*, bisa memilih warna dengan *color picker*.
5. Berbasis .NET, penjalanan program dilakukan dengan *Open Source* .NET *platform*, Mono. [11]

2.7 Bahasa Pemrograman C#

C# merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka .NET *Framework*. C# adalah *Java* versi Microsoft, sebuah bahasa *multi platform* yang didesain untuk bisa berjalan di berbagai mesin. C# adalah pemrograman berorientasi Objek (OOP). C# memiliki kekuatan bahasa C++ dan portabilitas seperti *Java*. Fitur-fitur yang diambilnya dari bahasa C++ dan *Java* adalah desain berorientasi objek, seperti *garbage collection*, *reflection*, akar kelas (*root class*), dan juga penyederhanaan terhadap pewarisan jamak (*multiple inheritance*).

Bahasa pemrograman C# dibuat sebagai bahasa pemrograman yang bersifat *general-purpose* (untuk tujuan jamak), berorientasi objek, modern, dan sederhana. C# ditujukan agar cocok digunakan untuk menulis program aplikasi baik dalam sistem klien-server (*hosted system*) maupun sistem *embedded* (*embedded system*), mulai dari program aplikasi yang sangat besar yang menggunakan sistem operasi yang canggih hingga kepada program aplikasi yang sangat kecil.

Meskipun aplikasi C# ditujukan agar bersifat 'ekonomis' dalam hal kebutuhan pemrosesan dan memori komputer, bahasa C# tidak ditujukan untuk bersaing secara langsung dengan kinerja dan ukuran program aplikasi yang dibuat dengan menggunakan bahasa pemrograman C.

2.8 *Adobe Photoshop*

Adobe Photoshop, atau biasa disebut *Photoshop* adalah perangkat lunak editor citra buatan *Adobe System* yang dikhususkan untuk pengeditan foto/gambar dan pembuatan efek. Perangkat lunak ini banyak digunakan oleh fotografer digital dan perusahaan iklan sehingga dianggap sebagai pemimpin pasar (marketleader) untuk perangkat lunak pengolah gambar/foto, dan bersama *Adobe Acrobat* dianggap sebagai produk terbaik yang pernah diproduksi oleh *Adobe System*. Versi kedelapan aplikasi ini disebut dengan nama *Photoshop CS* (*Creative Suite*), versi kesembilan disebut *Adobe Photoshop CS2*, versi kesepuluh disebut *Adobe Photoshop CS3*, versi kesebelas disebut *Adobe Photoshop CS4*, versi keduabelas disebut *Adobe Photoshop CS5*, dan versi yang terakhir (ketigabelas) adalah *Adobe Photoshop CS6*. [12]

Photoshop tersedia untuk *Microsoft Windows*, *Mac OS X*, dan *Mac OS*, versi 9 ke atas juga dapat digunakan oleh sistem operasi lain seperti *Linux* dengan bantuan perangkat lunak tertentu seperti *CrossOver*.

Meskipun pada awalnya *Photoshop dirancang* untuk menyunting gambar untuk cetakan berbasis-kertas, *Photoshop* yang ada saat ini juga dapat digunakan untuk memproduksi gambar untuk *World Wide Web*. Beberapa versi terakhir juga menyertakan aplikasi tambahan, *Adobe ImageReady*, untuk keperluan tersebut. *Photoshop* juga memiliki hubungan erat dengan beberapa perangkat lunak penyunting media, animasi, dan *authoring* buatan-*Adobe* lainnya. File format asli *Photoshop*, .PSD, dapat diekspor ke dan dari *Adobe ImageReady*. *Adobe Illustrator*, *Adobe Premiere Pro*, *After Effects* dan *Adobe Encore* DVD untuk membuat DVD profesional, menyediakan penyuntingan gambar non-linear dan layanan *special effect* seperti background, tekstur, dan lain-lain untuk keperluan televisi, film, dan

situs web. Sebagai contoh, *Photoshop CS* dapat digunakan untuk membuat menu dan tombol (button) DVD