BAB 2

LANDASAN TEORI

2.1 *Kecerdasan Buatan (Artificial Intelligence)*

Kecerdasan Buatan (*Artificial Intelligence*) merupakan salah satu bagian dari ilmu komputer yang mempelajari bagaimana membuat mesin (*computer*) dapat melakukan pekerjaan seperti sebaik yang dilakukan manusia bahkan lebih baik daripda yang dilakukan manusia.

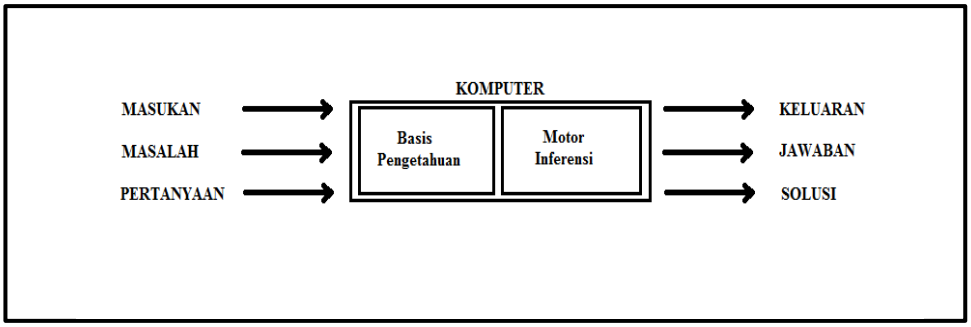
Menurut John McCarthy, 1956, AI: untuk mengetahui dan memodelkan proses-proses berpikir manusia dan mendesain mesin agar dapat menirukan prilaku manusia. Cerdas, berarti memiliki pengetahuan ditambah pengalaman, penalaran (bagaimana membuat keputusan dan mengambil tindakan), moral yang baik.

Manusia pandai (cerdas) dalam menyelesaikan permasalahan karena manusia mempunyai pengetahuan dan pengalaman.Pengetahuan diperoleh dari belajar.Semakin banyak bekal pengetahuan yang dimiliki tentu akan lebih mampu dalam menyelesaikan permasalahan. Tapi bekal pengetahuan saja tidak cukup, manusia juga diberi akal untuk melakukan penalaran, mengambil kesimpulan dengan pengalaman dan pengetahuan yang dimiliki.

Demikian juga agar mesin bisa cerdas (bertindak seperti dan sebaik manusia gambar 2. 1) maka harus diberi bekal pengetahuan, sehingga mempunyai kemampuan untuk menalar . Untuk membuat aplikasi kecerdasan buatan ada 2 bagian utama yang sangat dibutuhkan:

1. Basis Pengetahuan (*Knowledge Base*), bersifat fakta-fakta, teori, pemikiran dan hubungan antar satu dengan yang lainya.
2. Motor Infrensi (*Infrensi Engine*), kemampuan menarik kesimpulan berdasarkan pengetahuan dan pengalaman[4].

   Penerapan konsep kecerdasan buatan pada komputer adalah:



Gambar 2 Penerapan konsep kecerdasan buatan pada komputer

2.2 *Algoritma Fuzzy Logic*

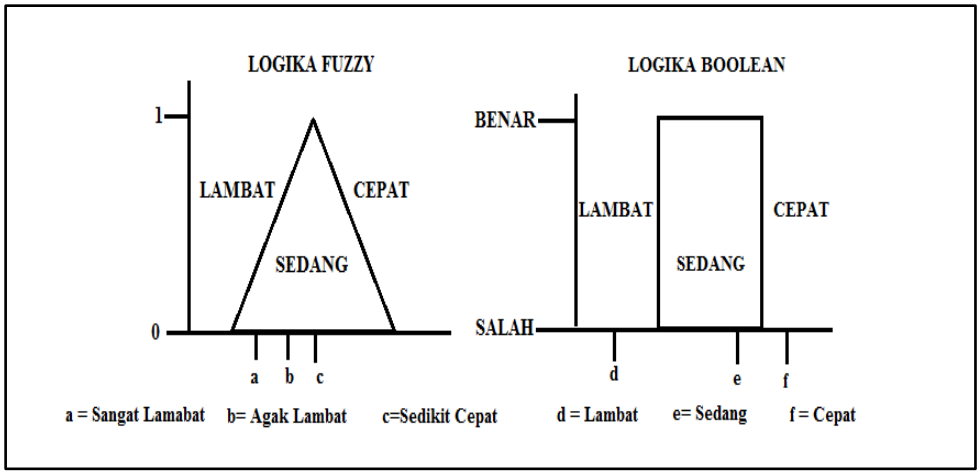
Logika *fuzzy* adalah peningkatan dari logika Boolean yang berhadapan dengan konsep kebenaran sebagian. Dimana logika klasik (crisp) menyatakan bahwa segala hal dapat diekspresikan dalam istilah binary (0 atau 1, hitam atau putih, ya atau tidak). Logika fuzzy menggantikan kebenaran Boolean dengan tingkat kebenaran. Logika fuzzy memungkinkan nilai keanggotaan antara 0 dan 1, tingkat keabuan dan juga hitam dan putih, dan dalam bentuk *linguistic*, konsep tidak pasti seperti “sedikit”, “lumayan”, dan “sangat”.

Logika ini diperkenalkan oleh Dr.Lotfi Zadeh dari Universitas California, Barkeley pada tahun 1965. Logika *fuzzy* telah digunakan pada bidang-bidang seperti taksonomi, topologi, linguistik, teori automata, teori pengendalian, psikologi, pattern recogniti on, pengobatan, hukum, decision analysis, system theory and information retrieval.Pendekatan fuzzy memiliki kelebihan pada hasil yang terkait dengan sifat kognitif manusia, khususnya pada situasi yang melibatkan pembentukan konsep, pengenalan pola, dan pengambilan keputusan dalam lingkungan yang tidak pasti atau tidak jelas[5].

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 dibawah ini.



Gambar 2.1 Pendefinisian kecepatan dalam bentuk *fuzzy logic*

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 menunjukan ukuran dari suatu variable seperti : 40, 25, 50, dsb.

2.2.2 *Fungsi – Fungsi Keanggotaan*

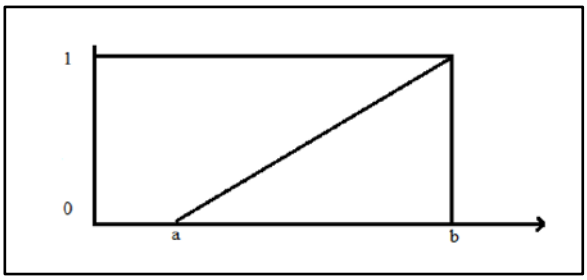
Didalam *fuzzy system*, fungsi keanggotaan memainkan peranan yang sangat penting untuk merepresentasikan masalah dan menghasilkan keputusan yang akurat. Terdapat banyak sekali fungsi keanggotaan yang biasa digunakan. Disini hanya membahas beberapa fungsi keanggotaan yang sering digunakan di dunia nyata, yaitu :

1. Representasi Linear

Pada representasi linear, pemetaan input ke derajat keanggotaan digambarkan sebagai suatu garis lurus.Bentuk ini paling sederhana dan menjadi pilihan yang paling baik utuk mendekati suatu konsep yang kurang jelas ada 2 keadaan himpunan *fuzzy* yang linear diantaranya:

Fungsi Keanggotaan Liner Naik:

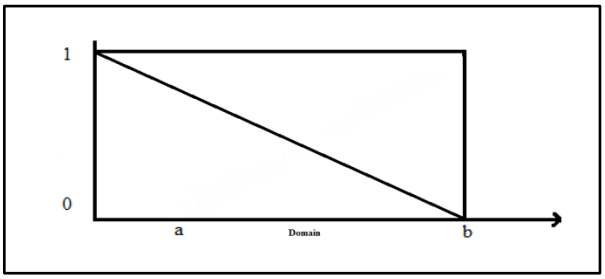
$$\mu[x] = \begin{cases} 0, x \leq a \\ \frac{b-x}{b-a}, a \leq x \leq b \\ 1, x \geq b \end{cases} \quad (2.1)$$



Gambar 2.2 Representasi Linear Naik

Fungsi Keanggotaan Linear Turun.

$$\mu[x] = \begin{cases} \frac{x-a}{b-a}, a \leq x \leq b \\ 0, x \geq b \end{cases} \quad (2.2)$$

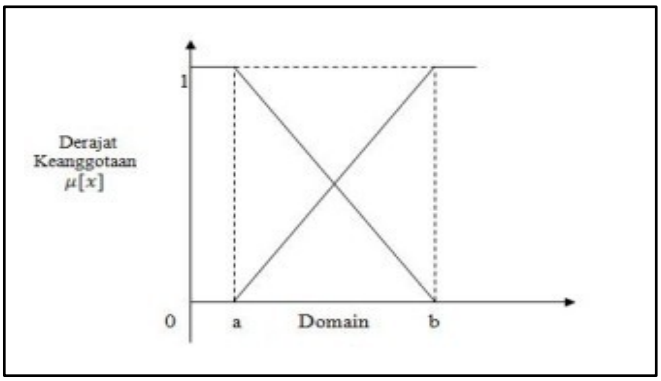


Gambar 2.3 Representasi Linear Naik

2. Fungsi Bahu

Daerah yang terletak di tengah-tengah suatu variabel yang direpresentasikan dalam bentuk segitiga, pada sisi kanan dan kirinya akan naik turun. Tetapi terkadang salah satu sisi dari variabel tersebut tidak mengalami perubahan. Himpunan fuzzy “bahu”, bukan segitiga, digunakan untuk mengakhiri variabel suatu daerah fuzzy. Bahu kiri bergerak dari benar ke salah, demikian juga bahu kanan bergerak dari salah ke benar. dapat dilihat pada Gambar 2.4

$$\text{Fungsi Bahu } (x,a,b) = \begin{cases} \mathbf{0}, & \boldsymbol{x \le b} \\ \left(\frac{b-x}{b-a}\right), & \boldsymbol{a \le x \le b} \\ \mathbf{1}; & \boldsymbol{x \ge a} \\ \mathbf{0}; & \boldsymbol{x \le a} \\ \left(\frac{x-a}{b-a}\right), & \boldsymbol{a \le x \le b} \\ \mathbf{1}; & \boldsymbol{x \ge b} \end{cases} \quad (2.3)$$

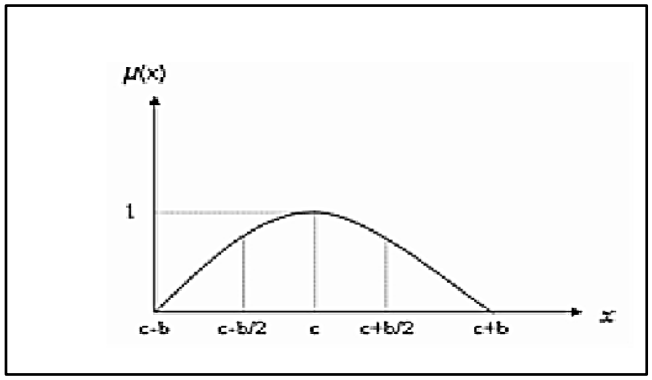


Gambar 2.4 Grafik dan notasi fungsi kurva bahu

3. 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 2.5

$$\text{Phi}(x,b,c) = \begin{cases} \boldsymbol{Phi\left(x, c-b, c-\frac{b}{c}, c\right), x \le c} \\ \boldsymbol{1 - phi\left(x, c, c+\frac{b}{2}, c+b\right), x > c} \end{cases} \quad (2.4)$$

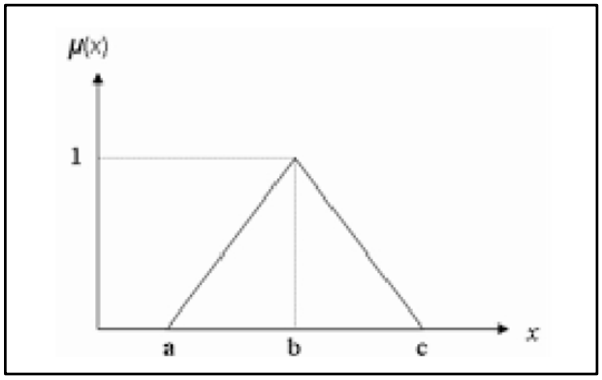


Gambar 2.5 Grafik dan notasi fungsi phi

4. Fungsi 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 2.6

$$\text{Segitiga}(x,a,b,c) = \begin{cases} \mathbf{0}, & \boldsymbol{x \leq a \; atau \; x \geq c} \\ \boldsymbol{(x-a)/(b-a)}, & \boldsymbol{a < x \leq b} \\ \boldsymbol{(c-x)/(c-b)}, & \boldsymbol{b < x \leq c} \end{cases} \qquad (2.5)$$

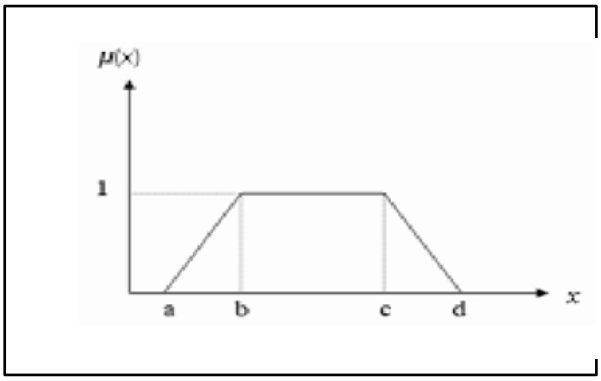


Gambar 2.6 Garfik dan notasi fungsi segitiga

5. Fungsi trapesium

Berbeda dengan fungsi segitiga, pada fungsi ini terdapat beberapa nilai x yang memiliki derajat keanggotaan sama dengan 1, yaitu ketika b≤ x≤ c. Tetapi derajat keanggotaan untuk a< x <b dan c< x≤ d memiliki karakteristik yang sama dengan fungsi segitiga. Grafik dan notasi matematika untuk fungsi ini dapat dilihat pada Gambar 2.7

$$\text{Trapesium}\mu[x] = \begin{cases} \mathbf{0}, \boldsymbol{x \leq a \; atau \; x \geq d} \\ \boldsymbol{\frac{x-a}{b-a}}, \boldsymbol{a < x \leq b} \\ \mathbf{1}, \boldsymbol{b \leq x \leq c} \\ \boldsymbol{\frac{d-x}{d-c}}, \boldsymbol{c < x \leq d} \end{cases} \qquad (2.6)$$



Gambar 2.7 Garfik dan notasi fungsi

2.2.3 Variabel *Linguistik*

Suatu himpunan *fuzzy* bisa didefiniskan berdasarkan variable linguistic tertentu. Variabel l*inguistic* didefinisikan sebagai (u,T(u),*U*,R,S) dengan *U* adalah nama variable *linguistic*; T(u) adalah himpunan term*(linguistic value/linguistic* label) pada u dan masing-masing term didefinisikan dengan fungsi keanggotaan yang normal (mempunyai harga maksimum sama dengan 1) dan convex pada *U*; R adalah aturan sintatik untuk menghasilkan nama nilai-nilai pada u; dan S adalah aturan semantic untuk menghubungkan tiap nilai dengan artinya.

1. *Fuzzyfication*

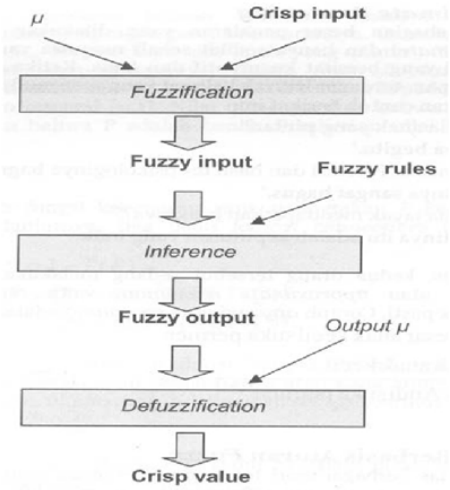
Mengubah masukan - masukan yang nilai kebenarannya bersifat pasti (*Crips input*) kedalam bentuk *fuzzy input*, yang berupa nilai linguistik yang semantiknya ditentukan berdasarkan fungsi keanggotaan tertentu.

2. *Inference*

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

3. *Defuzzyfication*

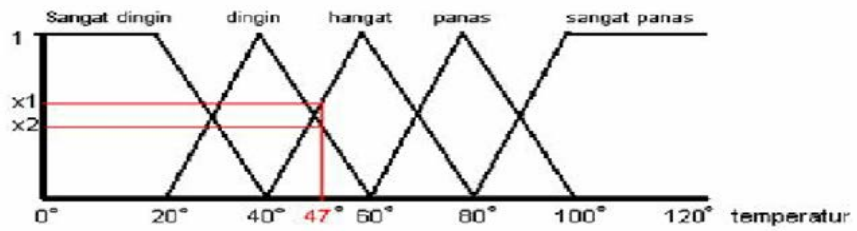
Mengubah *fuzzy output* menjadi crisp *value* berdasarkan fungsi keanggotaan yang telah ditentukan.



Gambar 2.8 diagram blok untuk sistem berbasis aturan *fuzzy* [5].

2.2.4 *Fuzzyfikasi*

Fuzzyfikasi yaitu suatu proses untuk mengubah suatu masukan dari bentuk tegas (crips) menjadi fuzzy (variable *linguistic*) yang biasanya disajikan dalam bentuk himpunan-himpunan fuzzy dengan suatu fungsi keanggotaan masing-masing . Contoh dari proses *Fuzzyfication* adalah seperti yang ditunjukan di gambar 2.9. Sebuah sistem fuzzy untuk mengukur suhu mempunyai 5 buah membership function yang mempunyai label sangat dingin,dingin, hangat, panas, sangat panas. Seperti pada gambar 2.9



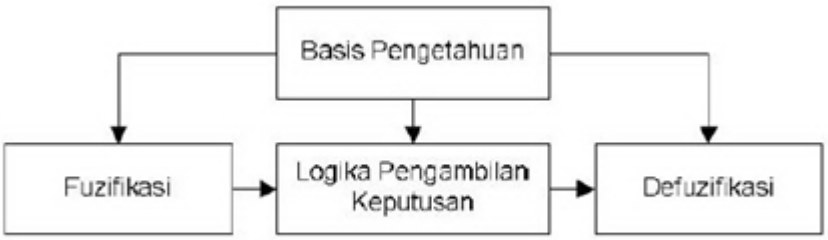
Gambar 2.9 Proses perubahan dari *crisp* input menjadi *fuzzy input*

2.2.5 *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 *x1, ..., xn* dan *m* variabel tidak bebas y1,...,ym. Misalkan *R* adalah suatu basis dari sejumlah *r* aturan fuzzy.

IF *P1*(x1,...,*xn*) THEN *Q1* (*y1*,...,*ym*), IF *Pr*(x1,...,*xn*) THEN *Qr* (*y1*,...,*ym*),

Dimana *p1,...pr* menyatakan fuzzy predicate untuk variabel bebas, dan *Q1,...Qr* menyatakan fuzzy predicate untuk variabel tidak bebas. Struktur sistem inferensi *fuzzy* dapat dilihat pada gambar 2.10



Gambar 2.10 Struktur sistem inferensi fuzzy [5].

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 tiga model aturan *fuzzy* yang digunakan secara luas dalam berbagai aplikasi.

1. 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: $f(x_1,\ldots,x_n) = w_0 + w_1x_1 + \ldots+w_nx_n$ dimana w_0, $w_1,\ldots,w_n$ adalah konstanta yang berupa bilangan real yang merupakan bagian dari spesifikasi aturan fuzzy.

2. Model Tsukamoto

Pada dasarnya, metode Tsukamoto mengaplikasikan penalaran monoton pada setiap aturanya . Kalau pada penalaran monoton , sistem hanya memiliki satu aturan , pada metode Tsukamoto , sistem terdiri atas beberapa aturan. Karena menggunakan konsep dasar penalaran monoton , pada metode Tsukamoto , setiap konsekuen pada aturan yang berbentuk IF-THEN harus direpresentasikan dengan suatu himpunan fuzzy dengan fungsi keanggotaan yang monoton . Output hasil inferensi dari tiap-tiap aturan diberikan secara tegas (crips) berdasarkan α predikat (fire strength). Proses agregasi antar aturan dilakukan, dan hasil akhinya diperoleh dengan menggunakan defuzzy dengan konsep rata-rata terbobot[6]. Misalkan ada variable masukan , yaitu x dan y , serta satu variable keluaran yaitu z. Variabel x terbagi atas 2 himpunan yaitu A1 dan A2 , variable y tebagi atas 2 himpunan yaitu B1 dan B2 , sedangkan variable keluaran z terbagi atas 2 himpunan yaitu C1 dan C2. Tentu saja himpunan C1 dan C2 harus merupakan himpunan yang bersifat monoton. Diberikan 2 aturan sebagai berikut:

[R1] IF x is A1 and y is B2 THEN z is (C1)

[R2] IF x is A2 and y is B1 THEN z is (C2)

Berdasarkan aturan *fuzzy* tersebut, akan ditentukan nilai α dan zuntuk masing-masing aturan. α adalah nilai keanggotaan anteseden dari setiap aturan, sedangkan z adalah nilai perkiraan barang yang akan diproduksi dari setiap aturan. Berikut ini adalah langkah-langkah untuk mengkonversi sembilan aturan fuzzy tersebut sehingga diperolah nilai α dan z dari setiap aturan. Pencarian nilai Z pada contoh kasus proses produksi perusahaan berdasarkan α-predikat dan z* untuk setiap aturan dengan menggunakan fungsi MIN pada aplikasi fungsi implikasinya.

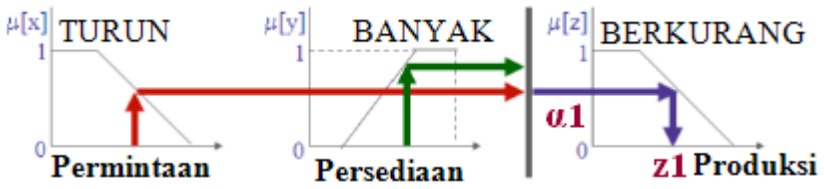
Berikut adalah contoh kasus metode tsukamoto dalam memetakan ruang input kedalam ruang output pada kasus menentukan jumlah produksi barang berdasarkan permintaan dan persediaan barang[12].

[R1] IF Permintaan TURUN And Persediaan BANYAK THEN Produksi Barang BERKURANG;

$\alpha - predikat1 = \mu pmt\text{TURUN} \cap \text{psdBANYAK}$

= min(μpmt TURUN [x], μpsd BANYAK [y])

Representasi aturan fuzzy [R1] ditunjukan pada gambar



Gambar 2.11 Representasi aturan *fuzzy* [R1].

z1 = zmax- α1(zmax-zmin) (2.7)

z1 adalah nilai z untuk aturan *fuzzy* [R1].

[R2] IF Permintaan TURUN And Persediaan SEDANG THEN Produksi Barang BERKURANG;

$\alpha - predikat2 = \mu\, pmt$TURUN $\cap$ psd SEDANG

= min(μpmt TURUN $x, \mu psd$ SEDANG $[y]$)

Representasi aturan fuzzy [R2] ditunjukan pada gambar



Gambar 2.12 Representasi aturan *fuzzy*[R2]

z2 = zmax - $\alpha 2$(zmax-zmin) (2.8)

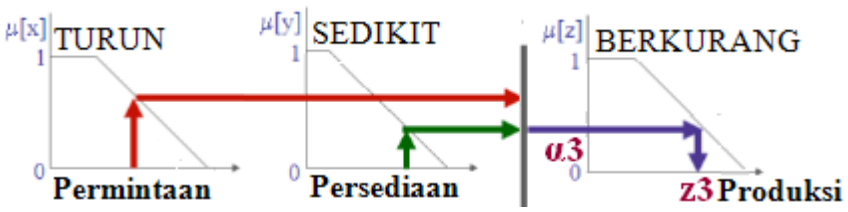
z2 adalah nilai z untuk aturan *fuzzy* [R2].

[R3] IF Permintaan TURUN And Persediaan SEDIKIT THEN Produksi Barang BERKURANG;

$\alpha - predikat3 = \mu\, pmt$TURUN $\cap$ psd SEDIKIT

= min(μpmt TURUN [x], μpsd SEDIKIT $[y]$)

Representasi aturan fuzzy [R3] ditunjukan pada gambar



Gambar 2.13 Representasi Aturan *Fuzzy*[R3]

z3 = zmax - α3(zmax-zmin) (2.9)

z3 adalah nilai z untuk aturan *fuzzy* [R3].

[R4] IF Permintaan TETAP And Persediaan BANYAK THEN Produksi Barang BERKURANG;

$\alpha - predikat4 = \mu\, pmt$TETAP $\cap$ psd BANYAK

= min(μpmt TETAP [x], μpsd BANYAK [y])

Representasi aturan fuzzy [R4] ditunjukan pada gambar



Gambar 2.14 Reperesentasi aturan *fuzzy*[R4]

Menurut fungsi keanggotaan produksi bertambah makamencari nilai z6 dengan

Rumus z4 = $zmax - \alpha$4 (zmax-zmin) (2.10)

z4 adalah nilai z untuk aturan *fuzzy* [R4].

[R5] IF Permintaaan TETAP And Persediaan SEDANG THEN Produksi Barang TETAP;

$\alpha - predikat5 = \mu\, pmt$TETAP $\cap$ psd SEDANG

= min(μpmt TETAP [x], μpsd SEDANG [y])

Representasi aturan fuzzy [R5] ditunjukan pada gambar



Gambar 2.15 Reperesentasi aturan *fuzzy*[R5]

$$\mu[x,y,z] = \begin{cases} 1: & x = xt \\ 1: & y = yt \\ 1: & z = zt \end{cases} \qquad (2.11)$$

z5=zt.

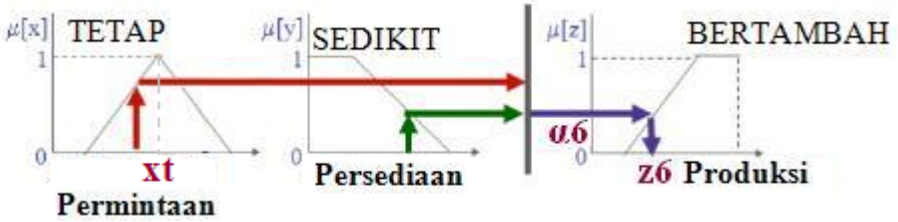
z5 adalah nilai z untuk aturan *fuzzy* [R5].

[R6] IF Permintaan TETAP And Persediaan SEDIKIT THEN Produksi Barang BERTAMBAH;

$\alpha - predikat6 = \mu\, pmt$TETAP $\cap$ psd SEDIKIT

= min(μpmt TETAP [x], μpsd SEDIKIT [y])

Representasi aturan fuzzy [R6] ditunjukan pada gambar



Gambar 2.16 Reperesentasi aturan *fuzzy*[R6]

z6= α6(zmax − zmin) + zmin (2.12)

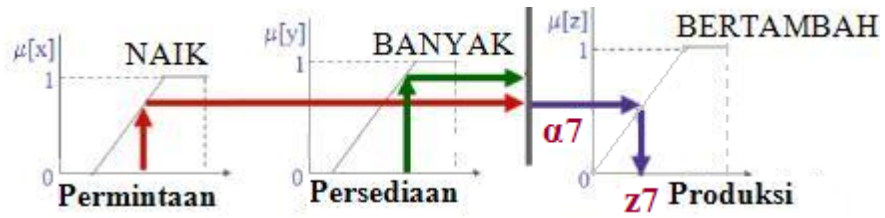
z6 adalah nilai z untuk aturan fuzzy [R6].

[R7] IF Permintaan NAIK And Persediaan BANYAK THEN Produksi Barang BERTAMBAH;

$\alpha - predikat7 = \mu pmt$NAIK ∩ psd BANYAK

= min(μpmt NAIK [x], μpsd BANYAK[y])

Representasi aturan fuzzy [R7] ditunjukan pada gambar



Gambar 2.17 Reperesentasi aturan *fuzzy*[R7]

z7= α7(zmax − zmin) + zmin (2.13)

z7 adalah nilai z untuk aturan fuzzy [R7].

[R8] IF Permintaan NAIK And Persediaan SEDANG THEN Produksi Barang BERTAMBAH;

$\alpha - predikat8 = \mu pmt$NAIK ∩ psd SEDANG

= min(μpmt NAIK [x], μpsd SEDANG[y])

Representasi aturan fuzzy [R8] ditunjukan pada gambar



Gambar 2.18 Reperesentasi aturan *fuzzy*[R8]

$z8 = \alpha 8(zmax - zmin) + zmin$ (2.14)

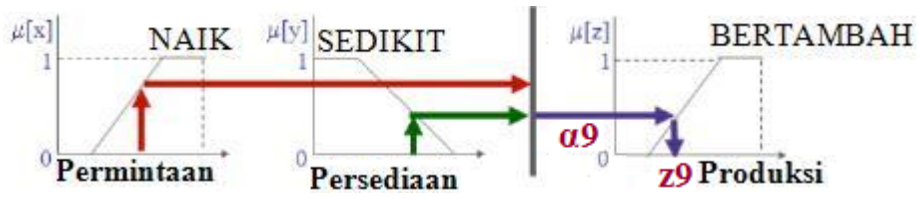
z8 adalah nilai z untuk aturan fuzzy [R8].

[R9] IF Permintaan NAIK And Persediaan Barang SEDIKIT THEN Produksi Barang BERTAMBAH;

$\alpha - predikat9 = \mu pmt\text{NAIK} \cap \text{psd SEDIKIT}$

$= \min(\mu pmt \text{ NAIK } [x], \mu psd \text{ SEDIKIT}[y])$

Representasi aturan fuzzy [R9] ditunjukan pada gambar



Gambar 2.19 Reperesentasi aturan *fuzzy*[R9]

$z9 = \alpha 9(zmax - zmin) + zmin$ (2.15)

z9 adalah nilai z untuk aturan fuzzy [R9].

Menentukan *Output Crips (Defuzzifikasi)*

$$Z = \frac{\alpha pred1 * z1 + \alpha pred2 * z2 + \alpha pred3 * z3 + \alpha pred4 * z4}{\alpha pred1 + \alpha pred2 + \alpha pred3 + \alpha pred4} \quad (2.16)$$

2.2.6 *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:[5]

1. Centroid method

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

$$\boldsymbol{y} * = \frac{\int \boldsymbol{y}\boldsymbol{\mu}(\boldsymbol{y})\boldsymbol{dy}}{\int \boldsymbol{\mu}(\boldsymbol{y})\boldsymbol{dy}} \qquad (2.17)$$

di mana y* suatu nilai crisp.
Fungsi integration dapat diganti dengan fungsi summation jika y bernilai diskrit, sehingga menjadi:

$$\boldsymbol{y} * = \frac{\sum \boldsymbol{y}\boldsymbol{\mu}(\boldsymbol{y})}{\sum \boldsymbol{\mu}(\boldsymbol{y})} \qquad (2.18)$$

di mana y* adalah nilai crisp dan (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

$$\boldsymbol{y} * = \frac{\boldsymbol{m+M}}{\boldsymbol{2}} \qquad (2.19)$$

m = nilai crisp yang paling kecil
M = nilai crisp yang paling besar.

5. Weighted Average

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

$$\boldsymbol{y} * = \sum \frac{\boldsymbol{\mu(y)y}}{\boldsymbol{\mu(y)}} \qquad (2.20)$$

y= nilai crisp

μ(y) = derajat keanggotan dari nilai crisp y.

2.2.7 *Operator Dasar Zadeh*

- Operator AND

Operator ini berhubungan dengan operasi interaksi pada himpunan. α-predikat sebagai hasil operasi dengan operator AND diperoleh dengan mengambil nilai keanggotaan terkecil antara elemen-elemen pada himpunan-himpunan yang bersangkutan.

$$\boldsymbol{\mu A \cap B = \min(\mu A[x], \mu B[y])} \qquad (2.21)$$

Rumus αi =zmax-zi / zmax-zmin

- Operator OR

Operator ini berhubungan dengan operasi union pada himpunan. α-predikat sebagai hasil operasi dengan operator AND diperoleh dengan mengambil nilai keanggotaan terkecil antar elemen pada himpunan-himpunan yang bersangkutan.

$$\mu AUB = \max(\mu A[x], \mu B[y]) \quad (2.22)$$

2.3 *Game*

Di dalam bagian ini akan dibahas tentang *game*, klasifikasi game berdasarkan platform,genre, serta sudut pandang permainan (*game* view point) dan berdasarkan pembatasan usia dari user pada *game* tersebut.

2.3.1 Pengertian *Game*

Game adalah sebentuk karya seni di mana peserta, yang disebut pemain, Membuat keputusan untuk mengelola sumber daya yang dimilikinya melalui benda di dalam *game* demi mencapai tujuan[13].

2.3.1.1 Video Game

Game elektronik atau selanjutnya dapat disebut video game pertama sekali ditemukan oleh Thomas T. Goldsmith dan Estle Ray Mann. Penemuan ini dipatenkan pada Januari 1947, yang mendasari perkembangannya saat itu adalah ketika mereka menemukan Cathode-Ray Tube sebuah tabung digunakan sebagai media untuk membuat simulasi kecepatan tembakan dan arah tembakan sebuah roket. Pada Februari 1951, Christopher Strachey memulai perkembangannya kearah pemrograman yang mulai menggunakan memori dimana aplikasinya diterapkan untuk kebutuhan para pilot. Dan penemuan baru terus berkembang hingga tahun 1959. Namun era perkembangan konsol game di mulai pada generasi pertama tahun 1972 sampai tahun 1977. Generasi kedua (1976-1982) sering disebut generasi konsol 8 bit. Di mana pemrograman video game dibuat lebih advance. Generasi Ketiga (1983-1992), Nintendo Entertainment System (NES) mendominasi pasarnya hingga Amerika Utara, Sega Master System juga berkembang pesat di Eropa dan

Brasil. Generasi Keempat (1987-1996) lebih dikenal dengan generasi 16 Bit dimana pengembangannya dimulai sekitar Oktober 1987. Generasi Kelima (1993-2002), konsol 64 bit mulai muncul pada era ini, grafis 3D mulai menjadi alasan untuk perang konsol ini. Generasi Keenam (1998-2006) merupakan generasi 128 bit, teknologi semakin maju membuat beberapa konsol baru mulai muncul dan semuanya memiliki pasar tersendiri. Generasi Ketujuh (2004) beberapa pengembang konsol mulai fokus dengan pengembangan saja. Mulai tahun 2004 hanya beberapa perusahaan konsol yang masih bertahan dengan pengembangannya Nintendo, Sony dan Microsoft.

2.3.1.2 Genre Pada Game

Menurut buku "*Fundamentals of Game Design*", genre *game* bisa dibagi menjadi 9 jenis, yaitu:

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. Vechicle Simulation

Vechicle simulation membuat *feeling* mengendarai kendaraan, baik kendaraan nyata maupun kendaraan imajiner. Performa dan 18 karakteristik kasar mesin harus menyerupai kenyataan, kecuali jika yang didesain adalah kendaraan imajiner.

6. *Construction and Management Simulation* (CMS)

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. 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.

8. Online Game

Istilah *online game* disini mengacu kepada *multiplayer game* dimana mesin dari para pemain terhubung dengan jaringan[7].

2.4 *Object Oriented Programing*

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)[8].

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[9].

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[9].

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.

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.

2.5 *UML (Unified Modelling Language)*

Unified Modelling Language (UML) 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.

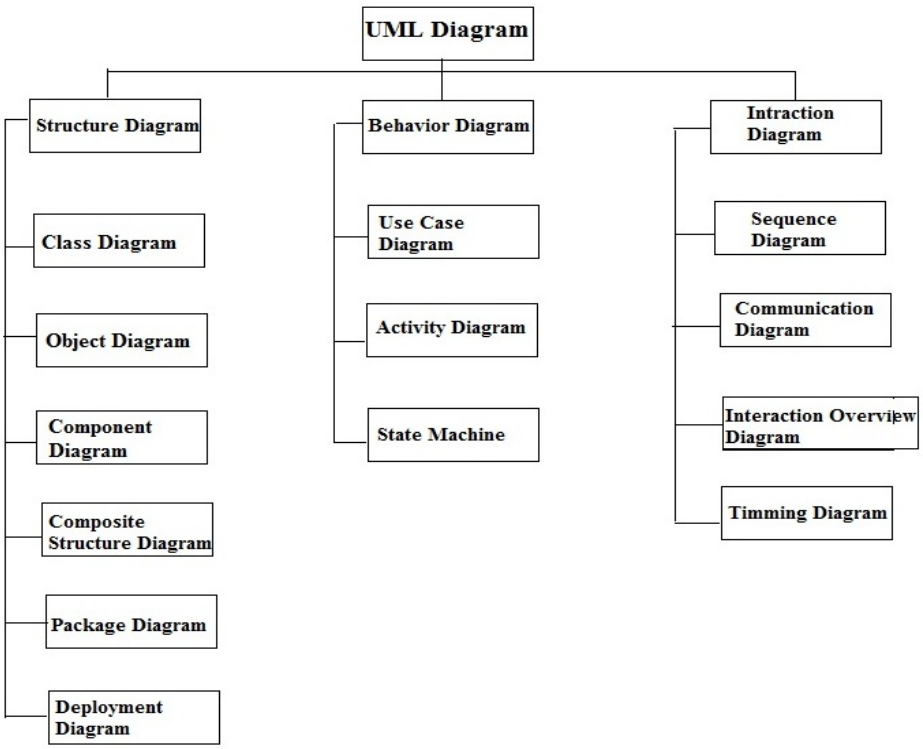
UML dapat digunakan untuk membuat model untuk semua jenis perangkat lunak, dimana perangkat lunak tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya,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

2.5.1 Diagram UML

Pada UML terdiri dari 13 macam diagram yang dikelompokan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar 2.20



Gambar 2.20 Diagram UML[9]

Berikut ini penjelasan singkat dari pembagian kategori tersebut:

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.

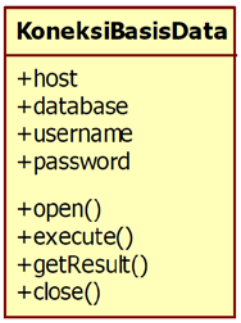
3. *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antarsubsistem pada suatu sistem.

2.5.2 Class Diagram

Diagram Kelas menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

1. Atribut merupakan variable-variabel yang dimiliki oleh suatu kelas.
2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas

Diagram kelas dibuat agar pembuat program membuat kelas-kelas sesuai rencana di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron.Berikut Contoh Class Diagram dapat dilihat pada Gambar 2.21



Gambar 2.21 Contoh *Class Diagram*[9]

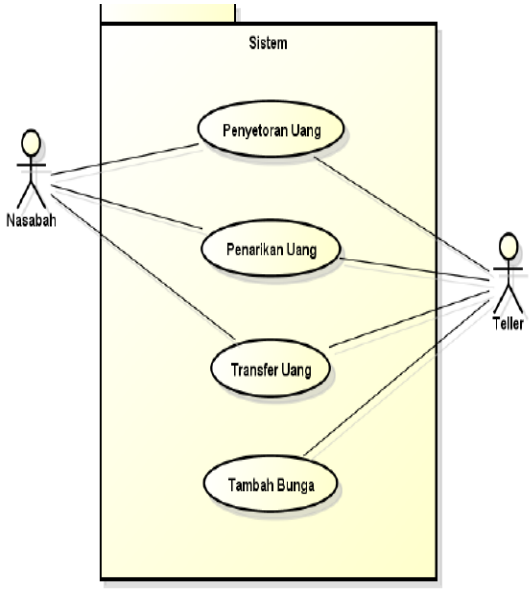
2.5.3 Diagram *Use Case*

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*[9].

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 unit-unit yang saling bertukar pesan antar unit atau aktor. Contoh Use Case Diagram dapat dilihat pada Gambar 2.22



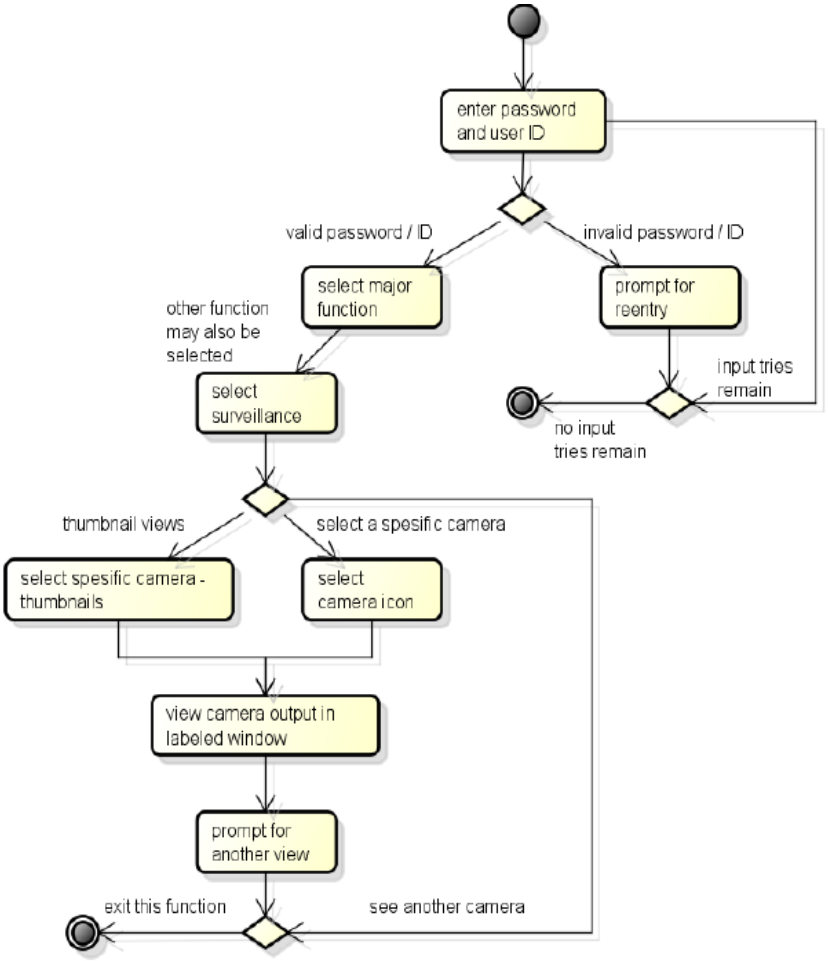
Gambar 2.22 Contoh *Use Case* Diagram

2.5.4 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:

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 2.23

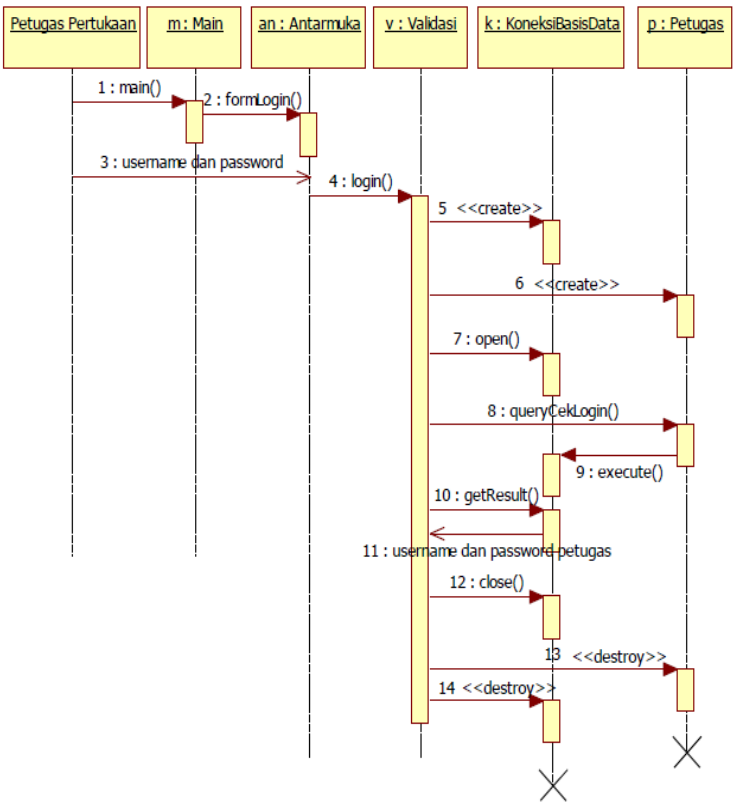


Gambar 2.23 Contoh *Activity Diagram*

2.5.5 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 2.24 [9].



Gambar 2.24 Contoh *Sequence Diagram*

2.6 *Adobe Flash*

Adobe Flash adalah *software* yang berlisensi yang dikeluarkan oleh *adobe system* yang sangat tenar dengan animasi gerak 2D atau 3D. Tipe file atau extensi dari adobe falsh adalah .swf. *Adobe flash* menggunakan bahasa pemrograman berjenis ActionScript, versi terbaru dengan ActionScript 3.0.

Flash pertama kali duluncurkan pada tahun 1996 dengan versi flash 1.0 kemudian Macromedia membeli animasi vektor FutureSplash Macromedia meluncurkan versi pertama sebelum 2005, Macromedia terakhir meluncurkan versi Macromedia Flash 8. Kemudian Flash berpindah tangan yang di akuisasi seluruh produk Macromedia pada Adobe System tanggal 3 Desember 2005 sehingga Macromedia Flash berubah jadi Adobe Flah[10].

2.6.1 *Adobe Flash Professional CS6*

Menurut buku Kupas Tuntas *Adobe Flash Professional CS6*, Adobe Flash Profesional perangkat lunak CS6 adalah lingkungan authoring kuat untuk menciptakan animasi dan konten multimedia. Desain pengalaman mendalam interaktif yang menyajikan secara konsisten di desktop dan beberapa perangkat, termasuk tablet, smartphone, dan televisi. Adobe Flash CS6 menyediakan berbagai macam fitur yang akan sangat membantu para animator untuk membuat animasi menjadi semakin mudah dan menarik. Adobe Flash CS6 telah mampu membuat dan mengolah teks maupun objek dengan efek tiga dimensi, sehingga hasilnya tampak lebih menarik.

Komponen *Adobe Flash Professional* CS6

1. *Create from Template* : berguna untuk membuka lembar kerja dengan template yang tersedia dalam program Adobe Flash CS6

2. *Open a recent item* : berguna untuk membuka kembali file yang pernah disimpan atau dibuka sebelumnya

3. *Create new* : berguna untuk membuka lembar kerja baru dengan beberapa pilihan script yang tersedia

4. *Learn* : berguna untuk membuka jendela Help yang berguna untuk mempelajari suatu perintah.

Komponen kerja Adobe Flash Professional CS6

1. *Toolbox*

   Toolbox adalah sebuah panel yang menampung tombol-tombol yang berguna untuk membuat suatu desain animasi mulai dari tombol seleksi, pen, pensil, text, 3D rotation, dan lain-lain.

2. *Timeline*

   Berguna untuk menentukan durasi animasi, jumlah layer, frame, menempatkan script dan beberapa keperluan animasi lainnya. Semua bentuk animasi yang di buat akan diatur dan ditempatkan pada layer dalam timeline

3. *Stage*

   Stage adalah lembar kerja yang digunakan untuk membuat atau mendesain objek yang akan dianimasikan. Objek yang dibuat dalam lembar kerja dapat berupa objek Vektor, Movie clip, Text, Button, dan lain-lain.

4. *Panel Properties*

   Panel properties berguna untuk menampilkan parameter dari sebuah tombol yang terpilih sehingga dapat dimodifikasi dan dimaksimalkan fungsi dari tombol tersebut. Panel properties menampilkan parameter sesuai dengan tombol yang terpilih

5. Efek *Filters*

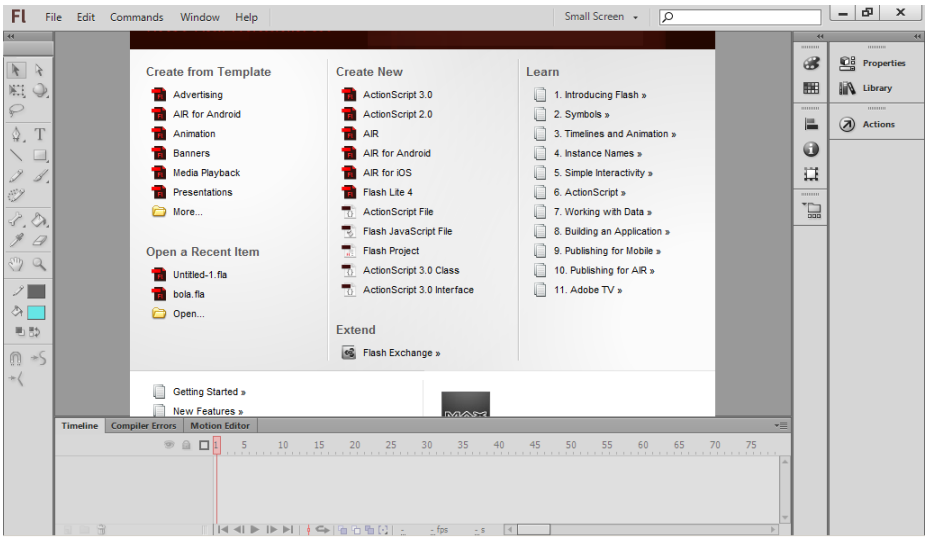
   Efek filters adalah bagian dari panel properties yang menampilkan berbagai jenis efek filter yang dapat digunakan untuk mempercantik tampilan objek. Filter hanya dapat diaplikasikan pada obejk Text, Movie clip dan Button.

6. *Motion Editor*

   Motion editor berguna untuk melakukan kontrol animasi yang telah dibuat, seperti mengatur motion, transformasi, pewarnaan, filter dan parameter animasi lainnya.

7. *Motion Presets*

   Panel motion presets menyimpan format animasi yang telah jadi dan siap digunakan sewaktu-waktu jika diperlukan. Ada berbagai pilihan animasi dalam panel motion presets, seperti sprila-3D, smoke, fly-out-top, dan lain-lain[11].



Gambar 2.25 Tampilan area kerja *adobe flash* CS6

2.7 *ActionScript 3.0*

ActionScript 3.0 atau disingkat AS3 merupakan bahasa pemrograman yang bekerja pada *Adobe Flash, Flex,* dan *FlashDevlop. ActionScript 3.0* pertama kali dirilis pada tahun 2006 bersamaan dengan diluncurkanya Flash versi 9 sekaligus Flash pertama yang kinitelah diakuisi oleh *Adobe System Inc*. yaitu *Flash* AS3.

ActionScript 3.0 adalah bhasa yang didasarkan pada ECMAScript, yaitu standar bahasa pemrograman yang dikembangkan oleh ECMA (Eouropan Computer Manufacturers Association). Dengan berbagi bahasa pemrograman lainya seperti JavaScript dan XML.

Dibandingkan dengan versi lainya ,versi AS3 dapat menghasilkan performa yang lebih baik dalam pembuatan game, namun semakin kompleks dari segi struktur bahasa. Ada sedikit perbedaan dari versi AS2 dan AS3 dalam penulisan script yaitu AS3 tidak bisa dituliskan langsung pada symbol movie clip ataupun button seperti AS2 sehingga kita bisa mengontrol movie dengan hanya menuliskan script pada frame atau melalui file kelas[1].