BAB II

LANDASAN TEORI

2.1 Bencana Banjir

Menurut Undang-undang nomor 24 Tahun 2007 tentang penanggulangan bencana, Bencana adalah peristiwa yang mengancam dan mengganggu kehidupan masyarakat disebabkan beberapa faktor yang mengakibatkan timbulnya korban jiwa manusia, kerusakan lingkungan, kerugian harta benda dan dampak psikologis. Bencana disebabkan tiga faktor yaitu faktor alam, faktor non alam dan faktor manusia. Jadi bencana dibagi menjadi tiga jenis yaitu bencana alam, bencana non alam dan bencana sosial. Bencana alam adalah bencana yang diakibatkan oleh peristiwa disebabkan oleh alam. Bencana yang disebabkan oleh alam seperti banjir, gempa bumi, tsunami, gunung meletus, kekeringan, angin topan dan tanah longsor (BNPB, Definisi dan Jenis Bencana, 2017).

Banjir adalah peristiwa terjadi genangan air di areal tertentu akibat dari meluapnya air sungai/danau/laut yang dapat menimbulkan kerugian baik materi maupun non-materi terhadap manusia dan lingkungan. (Jakarta & Jakarta, 2007). Banjir dalam pengertian umum adalah debit air yang secara relatif lebih besar dari kondisi normal akibat hujan yang turun di suatu tempat tertentu terjadi secara terus menerus, sehingga air tersebut tidak dapat ditampung, maka air melimpah keluar dan mengenangin daerah sekitarnya. (Sosial, 2009).

Penyebab banjir itu disebabkan beberapa faktor dan bila faktor tersebut dikelompokkan maka akan didapatkan tiga faktor yang mempengaruhi banjir yaitu elemen meteorologi, karakteristik fisik DAS dan manusia. Faktor meteorologi yang berpengaruh menimbulkan banjir adalah intensitas curah hujan, distribusi curah hujan dan frekuensi dan lamanya hujan berlangsung. Faktor karakteristik fisik DAS yang berpengaruh terhadap terjadi banjir yaitu luas DAS, kemiringan lahan, ketinggian dan kadar air tanah dan faktor manusia berperan pada percepatan perubahan karakteristik fisik DAS. (Suherlan, 2001).

Dampak akibat bencana banjir dapat dilihat dari tiga aspek yaitu sisi sosial, sisi ekonomi dan sisi lingkungan. Aspek sosial seperti terganggunya aktivitas masyarakat dan pemerintahan, aktivitas terkonsentrasi untuk mengatasi bencana banjir dan terganggunya kehidupan keluarga yang terkena banjir. Aspek ekonomi yaitu terganggunya aktivitas perekonomian, tergangggunya jalur distribusi dan terjadinya kerugian ekonomi. Aspek terakhir yaitu lingkungan seperti rusaknya infrastruktur, sarana dan prasana, rusaknya sanitasi lingkungan dan berjangkit penyakit pasca banjir seperti diare, penyakit kulit, infeksi saluran pernapasan akut (ISPA) dan lain-lain. (Perda Jakarta, 2009)

2.2 Curah Hujan

Menurut Badan Meterologi, Klimatologi, dan Geofisika (BMKG, Penjelasan Data Hujan, 2017), Hujan adalah titik-titik air di udara atau awan yang sudah telalu berat karena kandungan air sudah terlalu banyak, sehingga akan jatuh kembali ke

permukaaan bumi sebagai hujan (presipitas). Hujan dibagi menjadi 3 jenis dalam proses terjadi hujan yaitu

1. Hujan Orografis

   Hujan Orografis adalah hujan yang terjadi karena gerakan udara yang mengandung uap air terhalang oleh pengunungan sehingga massa udara itu dipaksa naik ke lereng pegunungan. Akitbanya suhu udara menjadi dingin. Sampai ketinggian tertentu terjadi proses kondensasi dan terbentuklah awan. Proses itulah terjadi hujan yang disebut dengan hujan orografis.

2. Hujan Konveksi (Zenithal)

   Hujan Konveksi adalah hujan terjadi karena udara yang mengandung uap air bergerak naik secara vertikal (konveksi) karena pemanasan. Udara yang naik itu mengalami penuruan suhu, sehingga pada ketinggian tertentu tejadi proses kondensasi dan pembentukan awan. Setelah awan tersebut tidak mampu menahan kumpulan titik- titik air maka terjadi hujan konveksi. Hujan konveksi terjadi di daerah tropis yang mempunyai intensitas penyinaran matahari yang selalu tinggi

3. Hujan Frontal

   Hujan frontal adalah hujan yang terjadi karena adanya pertemuaan antara massa udara panas dengan massa udara dingin. Pada pertemuan udara panas dan udara dingin terjadilah bidang *front* dimana terjadi kondensasi dan pembentukan awan. Udara yang panas selalu berada diatas udara

yang dingin. Hujan frontal biasanya terjadi di daerah lintang sedang atau pertengahan.

Menurut Badan Meterologi, Klimatologi, dan Geofisika (BMKG, Penjelasan Data Hujan, 2017), Curah hujan adalah ketebalan atau ketinggian air hujan yang terkumpul dalam tempat yang datar, tidak menguap, tidak meresap dan tidak mengalir. Curah hujan 1 milimeter artinya dalam luasan satu meter persegi pada tempat yang datar terapung air setinggi satu millimeter atau terapung air sebanyak satu liter. Satuan curah hujan yang dipakai di Indonesia yaitu millimeter (mm). Alat untuk mengukur curah hujan yaitu fluviometer. Selain itu fluviometer, curah hujan dapat diukur menggunakan alat atau instrumental penakar hujan jenis Hellman.

Penakar hujan jenis Hellman ini merupakan alat penakar hujan menggunakan *recording* atau mencatat secara manual atau sendiri. Biasanya alat ini dipakai di stasiun-stasiun BMKG untuk pengamatan udara permukaaan. Pengamatan dengan menggunakan alat ini dilakukan setiap hari pada jam-jam tertentu meskipun cuaca dalam keadaan baik atau sangat cerah. Alat ini mencatat jumlah curah hujan yang terkumpul dalam bentuk garis vertikal yang tercatat pada kertas pias dan alat ini memerlukan perawatan yang cukup dan intensif untuk menghindari kerusakaan yang sering terjadi pada alat ini (BMKG, Penjelasan Data Hujan, 2017).

Intensitas curah hujan merupakan besaran curah hujan yang jatuh dalam satuan waktu tertentu. Besaran intensitas curah hujan berbeda-beda biasanya disebabkan oleh lamanya hujan turun terjadi pada kurun waktu tertentu sedangkan luas daerah penyebaran hujan menunjukkan geografis curah hujan yang dapat diawali oleh suatu titik penangkar hujan (Sosrodarsono, 1993). BMKG sudah

mengklasifikasi intensitas curah hujan di wilayah Indonesia menjadi dua bagian yaitu hujan harian dan hujan per jam. Tabel 2.1 menampilkan klasifikasi curah hujan harian sebagai berikut:

Tabel 2.1 Klasifikasi curah hujan harian

Sumber: BMKG, 2010

Kriteria	Intensitas curah hujan per hari
Hujan ringan	5-20 mm/hari
Hujan sedang	20-50 mm/hari
Hujan lebat	50-100 mm/hari
Hujan sangat lebat	>100 mm/hari

Tabel 2.2 menampilkan klasifikasi curah hujan per jam sebagai berikut:

Tabel 2.2 Klasifikasi curah hujan perjam

Sumber: BMKG, 2010

Kriteria	Intensitas curah hujan per jam
Hujan ringan	1-5 mm/jam
Hujan sedang	5-10 mm/jam
Hujan lebat	10-20 mm/jam
Hujan sangat lebat	>20 mm/jam

2.3 Logika Fuzzy

Logika fuzzy adalah suatu komponen dari *soft computing* yang telah banyak diterapkan di berbagai bidang kehidupan. Logika fuzzy dapat membantu manusia untuk pengambilan keputusan yang semakin banyak kondisi yang menuntut adanya

keputusan yang tidak hanya bisa dijawab dengan ‘Ya’ atau ‘Tidak’. (Kusumadewi & Purnomo, Aplikasi Logika Fuzzy, 2010)

Pertama kali logika fuzzy dikenalkan oleh Prof. Lotfi A. Zadeh pada tahun 1965 dengan teori himpunan fuzzy. Pada teori ini, menyatakan bahwa peranan derajat keanggotaan sangat penting untuk menenetukan keberadaan elemen dalam suatu himpunan. Derajat keanggotaan atau membership menjadi ciri utama pada logika fuzzy (Kusumadewi & Purnomo, Aplikasi Logika Fuzzy, 2010).

Logika fuzzy dapat dianggap sebagai kotak hitam yang menghubungkan antar ruang *input* menuju ke ruang *output*. Kotak hitam berisi metode atau cara yang dapat digunakan untuk mengolah data *input* menjadi *output* dalam bentuk informasi yang baik. Logika fuzzy memiliki keunggulan sehingga menjadi alasan orang menggunakan logika fuzzy ini. Berikut adalah keunggulan dari logika fuzzy sebagai berikut (Kusumadewi & Purnomo, Aplikasi Logika Fuzzy Edisi 2, 2013):

1. Konsep logika fuzzy mudah mengerti.
2. Logika fuzzy sangat fleksibel.
3. Logika fuzzy memiliki toleransi terhadap data yang tidak tepat.
4. Logika fuzzy mempunyai kemampuan untuk memodelkan fungsi-fungsi non-linear yang sangat kompleks.
5. Logika fuzzy dapat membangun dan mengaplikasikan pengalaman para pakar secara langsung tanpa harus melalui proses pelatihan.
6. Logika fuzzy ini dapat bekerjasama dengan teknik-teknik kendali secara konvensial.

Ada beberapa hal yang diperlukan untuk memahami sistem fuzzy yaitu

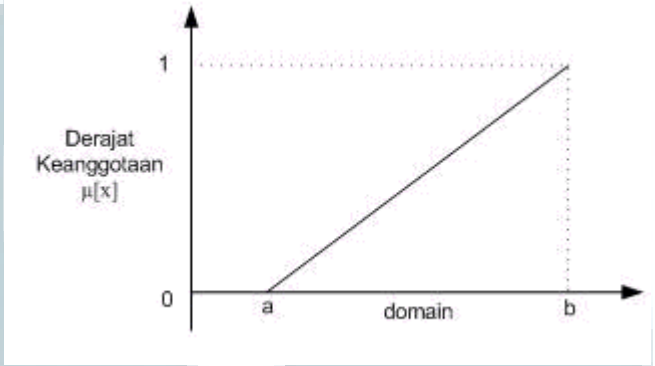
1. Variabel fuzzy adalah variabel yang hendak dibahas dalam suatu sistem fuzzy. Contoh: variabel umur, temperatur, permintaan dll.
2. Himpunan fuzzy adalah suatu grup yang mewakili suatu kondisi atau keadaaan dalam suatu variabel fuzzy. Contoh: variabel umur terbagi menjadi 3 himpunan fuzzy yaitu MUDA, PAROBAYA dan TUA.
3. Semesta Pembicaraan adalah suatu keseluruhan nilai yang diperboleh untuk operasi dalam suatu variabel fuzzy. Nilai semesta pembicaraan ini bisa berupa bilangan positif dan negatif. Contoh semesta pembicaraan untuk variabel umur yaitu (0-∞).
4. Domain adalah keseluruhan nilai yang diizinkan dalam semesta pembicaraan dan boleh dioperasikan dalam suatu himpunan fuzzy. Contoh domain himpunan fuzzy yaitu MUDA [0-45], PAROBAYA [35-55] dan TUA [45-∞].

Fungsi keanggotaan (*membership function*) adalah suatu kurva yang menunjukan pemetaan titik-titik input data ke dalam nilai keanggotaannya (sering disebut dengan derajat keanggotaan) yang memiliki interval antara 0 sampai 1. Beberapa fungsi yang bisa digunakan yaitu (Kusumadewi & Purnomo, Aplikasi Logika Fuzzy Edisi 2, 2013):

1. Representasi Linear

   Pada Representasi liniear, pemetaan *input* ke derajat keanggotannya digambarkan sebagai suatu garis lurus. Ada dua keadaan himpunan fuzy yang linear yaitu kenaikan himpunan dimulai pada nilai domain yang

memiliki derajat keanggotaan nol (0) bergerak ke kanan menuju ke nilai domain yang memiliki derajat kenggotaaan lebih tinggi. Gambar 2.1 menggambarkan representasi linear naik.



Gambar 2.1 Representasi linier naik

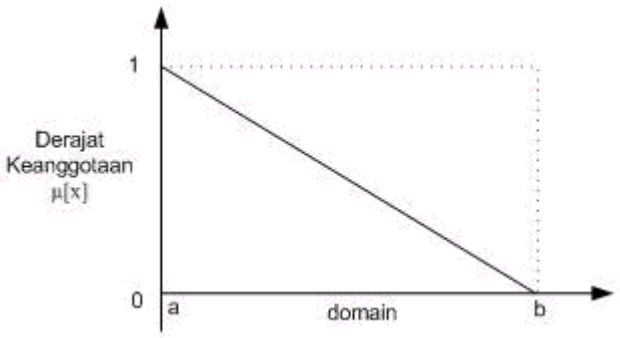
Sumber: Sri Kusumadewi, 2013

Rumus 2.1 menampilkan fungsi kenggotaan untuk representasi linier naik.

$$\mu[x] = \begin{cases} 0; & x \leq a \\ \dfrac{x-a}{b-a}; & a \leq x \leq b \end{cases}$$

Rumus 2.1 Fungsi keanggotaan untuk representasi linier naik

Kedua merupakan kebalikan yang pertama. Garis lurus mulai dari nilai domain dengan derajat keanggotaan tertinggi pada sisi kiri kemudian bergerak menurun ke nilai domain yang memiliki derajat keanggotaan lebih rendah. Gambar 2.2 menggambarkan representasi linear turun.



Gambar 2.2 Representasi linier turun

Sumber: Sri Kusumadewi, 2013

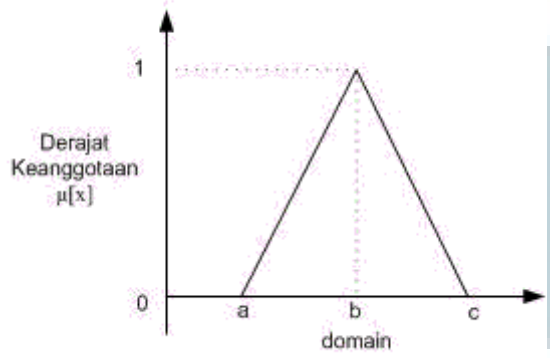
Rumus 2.2 menampilkan fungsi kenggotaan untuk representasi linier turun.

$$\mu[x] = \begin{cases} \frac{b-x}{b-a}; & a \le x \le b \\ 0; & x \ge b \end{cases}$$

Rumus 2.2 Fungsi keanggotaan untuk representasi linier turun

2. Representasi Kurva Segitiga

Kurva segitiga dasarnya merupakan gabungan antara 2 garis linier. Gambar 2.3 menggambarkan representasi kurva segitiga.



Gambar 2.3 Representasi kurva segitiga

Sumber: Sri Kusumadewi, 2013

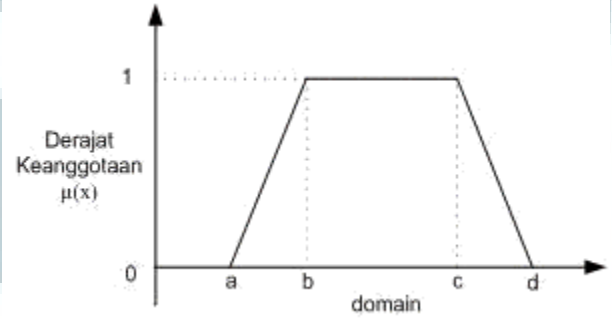
Rumus 2.3 menampilkan fungsi kenggotaan untuk representasi kurva segitiga.

$$\mu[x] = \begin{cases} 0; & x \leq a \; atau \; x \geq c \\ \dfrac{x-a}{b-a}; & a \leq x \leq b \\ \dfrac{b-x}{c-a}; & b \leq x \leq c \end{cases}$$

Rumus 2.3 Fungsi keanggotaan untuk representasi kurva segitiga

3. Representasi Kurva Trapesium

Kurva Segitiga pada dasarnya mirip bentuk kurva segitiga hanya saja ada beberapa titik yang memiliki nilai keanggotaan 1. Gambar 2.4 menggambarkan representasi kurva trapesium.



Gambar 2.4 Representasi kurva trapesium

Sumber: Sri Kusumadewi, 2013

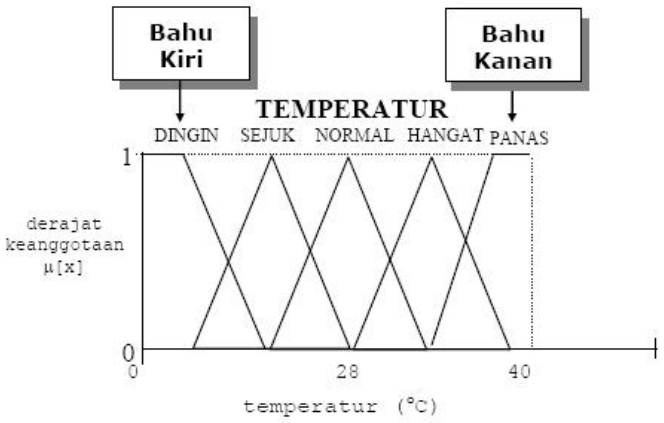
Rumus 2.4 menampilkan fungsi kenggotaan untuk representasi kurva trapesium:

$$\mu[x] = \begin{cases} 0; & x \leq a \; atau \; x \geq d \\ \dfrac{x-a}{b-a}; & a \leq x \leq b \\ 1 & b \leq x \leq c \\ \dfrac{d-x}{d-c}; & x \geq d \end{cases}$$

Rumus 2.4 Fungsi keanggotaan untuk representasi kurva trapesium

4. Representasi Kurva Bentuk Bahu

Daerah yang terletak di tengah suatu variabel yang direpresentasikan dalam bentuk segitiga, pada sisi kanan dan kirinya akan naik dan turun (Misal dari DINGIN bergerak ke SEJUK dan bergerak ke HANGAT dan bergerak ke PANAS). Tetapi terkadang satu sisi dari variabel tersebut tidak mengalami perubahaan. Misalnya sudah mencapai kondisi PANAS terjadi kenaikan suhu maka temperatur tetap pada kondisi PANAS karena Temperatur PANAS sudah temperatur paling tinggi. Bahu kiri bergerak dari benar ke salah sedangkan bahu kanan bergerak dari salah ke benar. Gambar 2.5 menggambarkan representasi kurva bentuk bahu untuk temperatur.



Gambar 2.5 Representasi kurva bentuk bahu pada variabel temperatur

Sumber: Sri Kusumadewi, 2013

Berikut operator himpunan fuzzy yang diciptakan oleh Zadeh berfungsi untuk mengkombinasi dan memodifikasi himpunan fuzzy. Nilai keanggotaan sebagai hasil dari operasi 2 himpunan sering dikenal dengan nama *fire strength* atau α-

predikat Berikut ada 3 operator dasar yang diciptakan oleh Zadeh yaitu (Kusumadewi & Purnomo, Aplikasi Logika Fuzzy Edisi 2, 2013):

1. Operator *AND* yaitu operator yang berhubungan dengan operasi interseksi atau beririsan pada himpunan. α-predikat adalah hasil dari operasi operator *AND* dan mengambil nilai keanggotaan terkecil antar elemen pada himpunan yang bersangkutan. Rumus 2.5 menampilkan rumus untuk operator *AND* sebagai berikut:

$$\mu_{A\cap B} = \min(\mu_a(x), \mu_b(y))$$

Rumus 2.5 Operator *AND*

2. Operator *OR* yaitu operator yang berhubungan dengan operasi *UNION* pada himpunan. α-predikat sebagai hasil dari operator *OR* dan mengambil nilai keanggotaaan terbesar antar elemen pada himpunan bersangkutan. Rumus 2.6 menampilkan rumus untuk operator *OR* sebagai berikut:

$$\mu_{A\cap B} = \max(\mu_a(x), \mu_b(y))$$

Rumus 2.6 Operator *OR*

3. Operator *NOT* yaitu operator yang berhubungan dengan operasi komplemen pada himpunan. α-predikat sebagai hasil dari operator *NOT* yang mengurangkan nilai keanggotaan elemen pada himpunan yang bersangkutan dari 1. Rumus 2.7 menampilkan rumus untuk operator *NOT* sebagai berikut:

$$\mu_{A'} = 1 - \mu_a(x)$$

Rumus 2.7 Operator *NOT*

Fungsi implikasi adalah tiap aturan atau *rule* pada basis pengetahuan fuzzy akan berhubungan dengan suatu relasi fuzzy. Rumus 2.8 menampilkan bentuk umum aturan yang digunakan dalam fungsi implikasi sebagai berikut:

IF x *is* A *THEN* y is b

Rumus 2.8 Aturan umum yang digunakan dalam fungsi implikasi

Dengan x dan y adalah skalar, dan A dan B yaitu himpunan fuzzy. Proposisi yang mengikuti IF disebut anteseden sedangkan THEN disebut dengan konsekuen. Pada fungsi implikasi mempunyai dua operator yaitu OR dan AND (Kusumadewi & Purnomo, Aplikasi Logika Fuzzy Edisi 2, 2013).

2.4 Sistem Inferensi Fuzzy Model Sugeno

Inferensi adalah penarikan kesimpulan. Jadi sistem inferensi fuzzy adalah penarikan kesimpulan dari sekumpulan kaidah fuzzy (Munir, 2011). Sistem inferensi fuzzy model sugeno diperkenalkan oleh Takagi-Sugeno Kang pada tahun 1985, sehingga metode ini sering disebut dengan metode TSK. Model Sugeno ada dua jenis yaitu (Kusumadewi & Purnomo, Aplikasi Logika Fuzzy Edisi 2, 2013):

1. Model Fuzzy Sugeno Orde-Nol

   Rumus 2.9 menampilkan secara umum bentuk model fuzzy Sugeno Orde-Nol adalah

   IF (x_1 is A_1) o (x_2 is A_2) o … o (x_n is A_n) *THEN* z = k

   Rumus 2.9 Model fuzzy Sugeno orde-nol

   Keterangan:

   A_i = himpunan fuzzy ke-i sebagai anteseden

k = konstanta (tegas) sebagai konsekuen

2. Model Fuzzy Sugeno Orde-Satu

   Rumus 2.10 menampilkan secara umum bentuk model fuzzy Sugeno Orde-Satu adalah

   IF (x_1 is A_1) o (x_2 is A_2) o … o (x_n is A_n) *THEN* $z = p_1*x_1+\ldots+p_n*X_n+q$

Rumus 2.10 Model fuzzy Sugeno orde-satu

Keterangan:

A_i = himpunan fuzzy ke-i sebagai anteseden

p_i = konstanta (tegas) ke-i

q = konstanta dalam konsekuen

Ada empat proses di dalam sistem inferensi fuzzy model sugeno yaitu *fuzzification*, *rule based*, *inference engine* dan *defuzzification*. Berikut penjelasan setiap proses di sistem inferensi fuzzy sebagai berikut:

1. *Fuzzification* adalah proses memetakan nilai *crip* (numerik) ke dalam himpunan fuzzy dan menentukan derajat keanggotaanya di dalam fuzzy. Hal ini dilakukan karena data diproses berdasarkan teori himpunan fuzzy sehingga data yang bukan dalam bentuk fuzzy harus diubah ke dalam bentuk fuzzy (Munir, 2011).
2. *Rule based* adalah suatu bentuk aturan relasi "*IF-THEN*". Aturan *IF-THEN* dihubungkan dengan operator yaitu *AND*, *OR* dan *NOT*. Rumus 2.11 menampilkan contoh *rule based* sebagai berikut: (Sari, 2015)

   IF x *is* a *AND* y *is* b *THEN* Z *is* c

Rumus 2.11 Contoh *rule based*

3. *Inference Engine* adalah proses implikasi dalam menalar nilai masukan untuk menentukan nilai keluaran sebagai pengambilan keputusan. Salah satu model penalaran yang banyak dipakai yaitu penalaran *max* dan *min* (Sari, 2015). Penalaran yang digunakan model sugeno yaitu *min* (Munir, 2011). Rumus 2.12 menampilkan contoh *inference engine* menggunakan penalaran *max* sebagai berikut:

var1 *is* A *OR* var2 *is* B *then max* (x1, x2) = x2

Rumus 2.12 Contoh *inference engine* menggunakan penalaran *max*

Rumus 2.13 menampilkan contoh *inference engine* menggunakan penalaran *min* sebagai berikut:

var1 *is* A *AND* var2 *is* B *then min* (x1, x2) = x1

Rumus 2.13 Contoh *inference engine* menggunakan penalaran *min*

4. *Defuzzification* adalah proses memetakan besaran dari himpunan fuzzy ke dalam bentuk nilai *crisp* (Munir, 2011). Pada sistem inferensi fuzzy model sugeno untuk deffuzifikasi dilakukan dengan cara mencari nilai rata-rata atau *weighted average*. Rumus 2.14 menampilkan rumus deffuzifikasi menggunakan nilai rata-rata atau *weighted average* adalah (Kusumadewi & Purnomo, Aplikasi Logika Fuzzy Edisi 2, 2013):

$$z = \frac{\alpha pred_1 * z_1 + \alpha pred_2 * z_2 + \cdots + \alpha pred_n * z_n}{\alpha pred_1 + \alpha pred_2 + \ldots + \alpha pred_n}$$

Rumus 2.14 Deffuzifikasi menggunakan *weighted average*

2.5 Metode *Exponential Smoothing*

Smoothing adalah mengambil rata-rata dari nilai pada beberapa periode untuk menaksir nilai pada suatu periode. *Exponential smoothing* adalah suatu metode peramalan rata-rata bergerak untuk melakukan pembobotan menurun secara *exponential* terhadap nilai observasi yang lebih tua. Jadi metode *exponential smoothing* merupakan pengembangan dari metode *moving average*. Dalam metode ini peramalan dilakukan dengan mengulang perhitungan secara terus menerus dengan menggunakan data terbaru. Metode *exponential smoothing* ada dua jenis yaitu (Santosa, Suharyanto, & Legono, 2010)

1. Metode *Single Exponential Smoothing*

   Metode *single exponential smoothing* merupakan perkembangan dari metode *moving average* sederhana yang mula-mula. Rumus 2.15 menampilkan rumus *forecast* menggunakan metode *single exponential smoothing* sebagai berikut:

$$f_{t+1} = \alpha X_t + (1 - \alpha)(f_t)$$

Rumus 2.15 *Forecast* menggunakan metode *single exponential smoothing*

   Keterangan:

   f_{t+1} = Peramalan data untuk satu jam mendatang

   α = Parameter *exponential* dari $0 < \alpha < 1$

   X_t = Data sekarang

   f_t = Peramalan data sebelumnya

2. Metode *Double Exponential Smoothing*

Metode *double exponential smoothing* merupakan model linier yang ditemukan oleh Brown. Didalam metode *double exponential smoothing* dilakukan proses *smoothing* dua kali. Metode *double exponential smoothing* lebih tepat untuk meramalkan data yang mengalamai tren kenaikan. Berikut ini proses-proses untuk *forecast* menggunakan *double exponential smoothing* yaitu

a. Menentukan *single exponential smoothing* (S'_t) dapat dilihat pada rumus 2.16

$$S'_t = \alpha X_t + (1 - \alpha)\,(S'_{t-1})$$

Rumus 2.16 *Single exponential smoothing*

b. Menentukan *double exponential smoothing*(S''_t) dapat dilihat pada rumus 2.17

$$S''_t = \alpha S'_t + (1 - \alpha)\,(S''_{t-1})$$

Rumus 2.17 *Double exponential smoothing*

c. Menentukan konstanta pemulus (α_t) dapat dilihat pada rumus 2.18

$$\alpha_t = 2S'_t - S''_t$$

Rumus 2.18 Konstanta pemulus

d. Menentukan nilai *slope* (b_t) dapat dilihat pada rumus 2.19

$$b_t = (\alpha / 1\text{-}\alpha) * (S'_t - S''_t)$$

Rumus 2.19 Nilai *slope*

e. Menentukan *forecast* dapat dilihat pada rumus 2.20

$$f_{t+1} = \alpha_t + b_{t*1}$$

Rumus 2.20 *Forecast* menggunakan metode *double exponential smoothing*

Keterangan:

$S't$ = Nilai *single exponential smoothing*

$S''t$ = Nilai *double exponential smoothing*

α = Parameter *exponential* dari $0 < \alpha < 1$

Xt = Data sekarang

St -1 = Peramalan data sebelumnya

at = Konstana pemulusan

b_t = *Slope*

f t+1 = Peramalan data untuk satu jam mendatang

2.6 Metode *Rapid Application Development* (RAD)

Rapid Application Development (RAD) adalah metode pengembangan perangkat lunak atau aplikasi yang berfokus pada membangun aplikasi dalam waktu yang singkat. RAD menekankan pada siklus pembangunan pendek, singkat dan cepat. Pengembangan dan perancangan perangkat lunak mengunakan metode RAD membutuhkan waktu sekitar 60-90 hari. RAD menggunakan metode iteratif (pengulangan) dalam pengembangan sistem dimana model kerja sistem dikonstruksikan di awal tahap pengembangan dengan tujuan untuk menetapkan kebutuhan pengguna atau *user* (Setiawan, Endrawan, Fathoni, & Budi, 2011). Metode RAD mempunyai tiga tahapan utama yaitu rencana kebutuhan

(*requirement planning*), proses desain (*design workshop*) dan implementasi (*implementation*). Berikut penjelasan dari tiga tahap utama pada metode RAD sebagai berikut (Noertjahyana, 2002):

1. Rencana kebutuhan (*Requirement Planning*)

   Pada tahap ini rencana kebutuhan yaiu mengidentifikasi tujuan dari pembuatan aplikasi ini dan mengidentifikasi kebutuhan informasi untuk mencapai tujuan aplikasi ini dibuat. Jadi tahap rencana kebutuhan ini sangat penting pada proses RAD supaya pengembangan aplikasi ini dapat mencapai tujuan dan menyelesaikan masalah-masalah yang ada.

2. Proses Desain (*Design Workshop*)

   Pada tahap ini yaitu mendesain proses yang berjalan di aplikasi atau mendesain alur logika di aplikasi supaya pengguna mengetahui proses aplikasi ini. Pada tahap desain ini membutuhkan waktu beberapa hari tergantung pada besar kecilnya aplikasi yang dibuat.

3. Implementasi (*Implementation*)

   Pada tahap ini yaitu mengembangkan sistem desain menjadi suatu aplikasi atau program. Setelah program selesai dibuat maka perlu proses pengujian untuk mengetahui apakah ada kesalahan atau program itu berjalan baik sebelum program itu diterapkan kepada pengguna atau *user*. Pada proses ini pengguna bisa memberikan tanggapan terhadap aplikasi yang dibuat serta persetujuan mengenai sistem tersebut.

2.7 Metode *Waterfall*

Metode *Waterfall* adalah metode pengembangan perangkat lunak yang paling tua dan paling banyak dipakai. Model ini mengusulkan pendekatan perkembangan perangkat lunak yang sistematik yang dimulai dari tahap analisis, desain, kode, pengujian dan pemeliharaan (Kurnia, Rachamadhani, & Sutarmaningtyas, 2012). Berikut penjelasan dari tahap-tahp pengembangan dan perancangan perangkat lunak menggunakan metode *waterfall* sebagai berikut (Kurnia, Rachamadhani, & Sutarmaningtyas, 2012):

1. Analisis kebutuhan perangkat lunak

   Pada tahap ini yaitu menganalisis dan mengumpulkan kebutuhan sistem yang sesuai dengan tujuan aplikasi yang dibuat. Kebutuhan tersebut dapat didokumentasikan atau dilihat oleh pengguna atau user.

2. Desain

   Pada tahap ini yaitu menerjemahkan syarat kebutuhan ke sebuah perancangan perangkat lunak. Proses ini berfokus pada struktur data, arsitektur perangkat lunak, representasi *interface* dan alur logika *procedural*.

3. Pengkodeaan (*Coding*)

   Pada tahap ini yaitu proses menerjemahkan desain ke dalam suatu bahasa yang bisa dimengerti oleh komputer.

4. Pengujian

   Pada tahap ini yaitu menguji aplikasi yang dibuat supaya tidak ada kesalahaan baik error atau bug pada aplikasi tersebut. Selain itu, aplikasi

ini diuji dengan memasukkan *input* dan akan memberikan hasil yang aktual sesuai yang dibutuhkan.

5. Pemeliharaan

   Perangkat yang sudah disampaikan kepada pelanggan pasti akan mengalami perubahan. Perubahan tersebut bisa karena mengalami kesalahan karena perangkat lunak harus menyesuaikan dengan lingkungan baru atau pelanggan membutukan perkembangan fungsional.

2.8 *Mobile Application*

Menurut (Turban, 2012) *mobile application* juga biasa disebut dengan *mobile apps*, yaitu istilah yang digunakan untuk mendeskripsikan aplikasi internet yang berjalan pada *smartphone* atau perangkat *mobile* lainnya. Aplikasi *mobile* biasanya membantu para penggunanya untuk terkoneksi dengan layanan internet yang biasa diakses pada PC atau mempermudah mereka untuk menggunakan aplikasi internet pada perangkat yang bisa dibawa.

Menurut (Jobe, 2013) *mobile application* dibagi menjadi beberapa jenis yaitu

1. *Native Application* (*Native App*)

   Native App adalah aplikasi yang dibuat dan dikembangkan untuk suatu sistem operasi mobile yang spesifik. Ada tiga sistem operasi mobile yang terbesar yaitu Android, iOS dan Windows Phone. Dalam pembuatan native app dapat menggunakan *Java Programming* untuk Android, *Objective C programming* untuk iOS dan .Net *framework* untuk Windows Phone. Karakteristik utama dari *native app* yaitu aplikasi ini

memiliki seluruh akses baik *hardware* atau *user interface* yang sesuai dengan jenis sistem operasinya.

2. *Mobile Web Application* (*Mobile Web App*)

   Mobile Web App adalah aplikasi *website* yang dirancang dan dikembangkan untuk meniru kemampuan *native app*. Aplikasi ini dapat dijalankan di *web browser* pada platform yang digunakan. *Mobile web app* dikembangkan dengan kombinasi HTML 5, Javascript dan CSS.

3. *Hybrid Mobile Application* (*Hybrid Mobile App*)

   Hybrid mobile app adalah aplikasi yang yang mengabungkan *native app* dan *web app*. *Hybrid mobile app* pada dasarnya dibuat dengan menggunakan teknik *web programming* seperti HTML 5, Javascript API dan CSS, tetapi bisa berjalan di 3rd *party native app container*. Karakteristik dari *hybrid mobile app* dalah aplikasi ini dapat dikembangkan dengan bahasa web standar tapi memiliki akses ke API *Native Apps* dan *hardware*. Ada beberapa framework *hybrid mobile apps* yang terkenal yaitu PhoneGap, Appcelerator, Appspresso dan Cordova.

2.9 SMS *Gateway*

Gateway dapat diartikan sebagai jembatan penghubung antar satu sistem dengan sistem berbeda, sehingga dapat terjadi pertukaran data antar sistem tersebut. Awalnya SMS *gateway* dibutuhkan untuk menjembatani antar SMS *Center* (SMSC). Dulu SMSC perusahaan memiliki protokol komunikasi sendiri, dan

protokol itu bersifat pribadi (Abuhari & IIyas, 2015). Jadi awalnya SMS *gateway* hanya menghubungan satu SMSC dengan SMSC yang lain.

Dengan seiring perkembangan teknologi komputer baik dari sisi *hardware* dan *software*, SMS *gateway* lebih mengarah pada sebuah perangkat lunak yang menggunakan bantuan komputer dan memanfaatkan teknologi seluler yang diintegrasikan untuk mendistribusikan pesan-pesan yang digenerate lewat sistem informasi melalui media SMS yang diatur oleh jaringan seluler (Prasetio, 2013). Jadi SMS gateway menghubungkan *software* atau aplikasi dengan perangkat komunikasi untuk mengirim atau menerima SMS.

Beberapa kemampuan SMS *gateway* sebagai berikut (Prasetio, 2013) :

1. Memperbesar skala aplikasi teknologi informasi dengan mengguakan komunikasi SMS interaktif.
2. Menyediakan aplikasi kolaborasi komunikasi SMS berbasis *web* untuk pengguna di perusahaan atau institusi
3. Menjangkau pelanggan atau pengguna jasa layanan perusahaan secara mudah untuk menggunakan komunikasi SMS interaktif.

Konsep dan cara kerja SMS *gateway* yaitu ketika pelanggan mengirim SMS ke sistem maka SMS akan masuk terlebih dahulu ke SMSC operator telepon yang digunakan. SMSC yaitu sebuah *server* yang bertanggung jawab pada proses pengiriman SMS pada suatu operator. Dari proses SMSC itu maka di ambil oleh Gammu dan dimasukan ke dalam tabel *inbox* dan selanjutnya di proses oleh PHP. Pesan yang diproses dinamakan oleh *autoreply*. *Autoreply* SMS akan di *query* oleh PHP kemudian dimasukan ke dalam tabel outbox. *Autoreply* pada tabel *outbox* akan

disalin ke dalam tabel *sent items*. *Autoreply* pada tabel *sent items* akan diambil oleh Gammu dan dikirim ke SMSC dan selanjutnya dikirim ke handphone pelanggan.

Pada sistem pengiriman SMS ke banyak nomor maka SMS yang dikirim tidak akan langsung dikirim ke nomor tersebut tetapi masuk terlebih dahulu ke SMS *Center* (SMSC), kemudian baru diteruskan ke nomor tujuan SMS tersebut. Bila nomor yang dituju mati atau *offline* maka SMSC akan menyimpan SMS tersebut untuk sementara waktu, hingga nomor yang dituju itu hidup atau *online*. Lama waktu penyimpanan SMS sangat tergantung dari lamanya waktu yang ditetapkan oleh operator untuk menyimpan SMS tersebut. Nomor yang telah menerima SMS akan mengirim laporan ke SMSC bahwa SMS telah diterima (Abuhari & IIyas, 2015).

Berikut beberapa fitur unggulan yang sering ditemukan pada SMS *Gateway* yaitu (Basuki, 2016):

1. *Auto Reply* / *Auto Responder*

   Fitur ini akan otomatis membalas SMS dari pengguna yang mengetik kode tertentu sesuai dengan format yang sudah diatur.

2. *Polling* SMS

   Poling SMS ini sering dipakai pada acara pencarian bakat di televisi. Ketika anda mendukung peserta tentu maka anda akan mengetik format tertentu untuk melakukan pemilihan dan polling.

3. *Broadcast Message*

   Fitur ini yang sering dipakai untuk melakukan pengiriman SMS ke banyak nomor tujuan sekaligus. Walaupun pada *smartphone* sekarang

sudah ada fitur ini, namun anda harus menambahkan nomor tujuan satu-persatu. Dengan aplikasi SMS *gateway* ini maka dapat diatur secara lebih baik.

4. *Scheduled Message*

   Fitur ini memungkinkan untuk mengirim SMS pada waktu yang sudah ditentukan sebelumya. Contoh ucapan selamat ulang tahun atau pesan pengingat.

2.10 *Class Diagram*

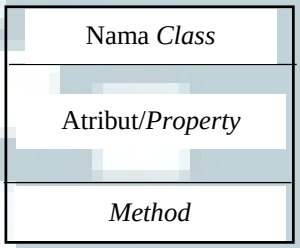
Class adalah kumpulan objek-objek yang mempunyai struktur, *behavior* dan relasi yang umum. *Class Diagram* adalah diagram yang menunjukan kelas-kelas yang ada dari sebuah sistem dan hubungannya secara logika. Menurut Whitten L. Jesffery, *class diagram* adalah gambar grafis mengenai struktur objek statis dari suatu sistem, menunjukan *class-class* objek yang menyusun sebuah sistem dan hubungan antar class objek tersebut (Henderi, 2009). Berikut adalah notasi dan simbol dasar pada elemen class diagram sebagai berikut:

1. Kelas / *Class* adalah himpunan dari objek-objek yang berbagi atribut serta operasi yang sama. Class dibagi menjadi tiga bagian yaitu nama class, atribut/ *property* dan metode yang dipakai class tersebut. Atribut dan metode memiliki salah satu sifat. Berikut sifat yang ada di atribut dan metode sebagai berikut (Dharwiyanti & Wahono, 2003):

   - *Private* adalah tidak bisa digunakan atau dipanggil dari luar class yang bersangkutan.

- *Protected* adalah hanya dapat dipanggil oleh class yang bersangkutan dan anak class yang mewarisinya.
- *Public* adalah dapat dipanggil oleh siapa saja.

Class dapat disimbolkan dalam bentuk kotak yang dibagi menjadi tiga bagian. Gambar 2.6 menampilkan simbol atau notasi dari *class* di *class diagram* sebagai berikut:



Gambar 2.6 Simbol *class* di *class diagram*

2. Relasi / *Relationship* adalah hubugan antar kelas yang memiliki *multiplicity*. *Multiplicity* menunjukan jumlah suatu objek yang bisa berhubungan dengan objek lain (Husien, 2013).

Relasi / *relationship* pada *class diagram* ada tiga yaitu asosiasi, generalisasi dan agregasi. Berikut ini adalah relasi yang ada di class diagram sebagai berikut (Dharwiyanti & Wahono, 2003):

1. Asosiasi adalah hubungan statis antar *class* yang berhubungan. *Class* yang mempunyai asosiasi bila *class* tersebut memiliki atribut dengan *class* lain. Asosiasi bisa berupa dua arah atau satu arah. Asosiasi dua arah disimbolkan dengan garis lurus sedangkan asosiasi satu arah disimbolkan dengan garis panah. Gambar 2.7 menampilkan simbol atau notasi dari asosiasi dua arah di *class diagram* sebagai berikut:

Gambar 2.7 Simbol asosiasi dua arah di *class diagram*

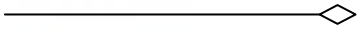
Gambar 2.8 menampilkan simbol atau notasi dari asosiasi satu arah di *class diagram* sebagai berikut:

Gambar 2.8 Simbol asosiasi satu arah di *class diagram*

2. Generalisasi adalah hubungan hirakis antar *superclass* dan *subclass*. *Superclass* yaitu adalah *class* yang mewarisi semua atribut dan metode ke *subclass*. *Subclass* adalah anak dari *class* yang diwarisin dari *superclass*. Contoh yaitu class karyawan memiliki hubungan generalisasi untuk class karyawan *part-time* dan karyawan tetap. Jadi *class* karyawan adalah *superclass* dan *class* karyawan *part-time* dan karyawan tetap yaitu *subclass*. Generalisasi disimbolkan dengan garis panah yang bentuk segitiga kosong. Gambar 2.9 menampilkan simbol atau notasi dari generalisasi di class diagram sebagai berikut:

Gambar 2.9 Simbol generalisasi di *class diagram*

3. Agregasi adalah hubungan antara keseluruhan dan bagian. Contoh relasi agregasi yaitu *class* mobil yang mempunyai bagian yaitu ban, mesin dan roda. Jadi *class* mobil mempunyai hubungan agregasi dengan *class* ban, mesin dan roda. Agregasi dapat disimbolkan dengan garis yang ujung berbentuk diamond kosong. Gambar 2.10 menampilkan simbol atau notasi dari agregasi di *class diagram* sebagai berikut:



Gambar 2.10 Simbol agregasi di *class diagram*

Relasi pada *class diagram* mempunyai *multiplicity* pada hubungan dan relasi antar class. *Multiplicity* adalah memberikan gambaran sejumlah instan yang akan ditampung dalam class. Tabel 2.3 menampilkan notasi *multiplicity* di class diagram sebagi berikut:

Tabel 2.3 Tabel notasi *mulipicity* di *class diagram*

Sumber: Sri Dharwiyanti, 2003

Multiplicity	**Arti**
0	Nol
1	Satu
*	Banyak
0..*	Nol atau lebih
1..1	Hanya satu
1.. *	Satu atau lebih

2.11 *Use Case Diagram*

Use case diagram adalah model fungsional pada sebuah sistem yang mengambarkan hubungan *actor* dan *use case* pada suatu sistem. *Use case diagram* dibuat untuk mengambarkan hubungan antara *actor* dan *use case* pada suatu sistem (Henderi, 2009). Tujuan u*se case diagram y*aitu untuk memfasilitasi komunikasi antar *user* dalam sistem dan klien dan menjelaskan apa yang harus diperbuat sistem

(Widianto, 2011). Berikut ini adalah elemen dasar pada *use case diagram* sebagai berikut:

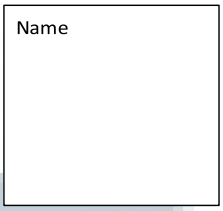
1. *Actor* adalah penguna atau *user* dari sebuah sistem. *Actor* tidak hanya *user* atau pengguna sistem tetapi sistem juga bisa menjadi *actor* bagi sistem lain. Jadi *Actor* adala seorang atau sesuatu yang harus berinterakasi dengan sistem atau sistem yang dibangun/ dikembangkan (Widianto, 2011). Gambar 2.11 menampilkan notasi atau simbol dari *actors* di *use case diagram* sebagai berikut:

Gambar 2.11 Simbol *actor* di *use case diagram*

2. *Use case* adalah sebuah aktivitas atau kegiatan yang dilakukan oleh *actor*. *Use case* disimbolkan dengan bentuk oval. *Use case* berisi kata kerja yang menggambarkan aktivitas yang dilakukan *actor* pada sistem tersebut. Gambar 2.12 mengambarkan notasi atau simbol dari *use case* sebagai berikut:

Gambar 2.12 Simbol *use case* di *use case diagram*

3. *Boundary* sistem adalah gambaran batasan sebuah sistem yang berisi *use case*. *Actor* yang terlibat pada *use case* tempatkan di bagian luar sistem boundries. *Boundary* sistem disimbolkan dalam bentuk persegi panjang yang berisi beberapa *use case* (Widianto, 2011). Gambar 2.13 menampilkan notasi atau simbol dari *boundary sistem* sebagai berikut:

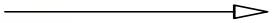


Gambar 2.13 Simbol *boundary* sistem di *use case diagram*

4. Relasi/*Relationship* adalah gambar relasi atau hubungan antara use case dan actor. Relasi terdapat empat tipe relasi yang digunakan pada use case diagram yaitu asosiasi, generalisasi, *extend* dan *include*. Berikut adalah penjelasan dari empat relasi yang terdapat di *use case diagram* sebagai berikut (Dzaky, 2017):
    a. Asosiasi digunakan untuk mengambarkan interaksi antara *actor* dengan setiap *use case* tertentu. Relasi ini digambarkan sebagai garis penghubung *actor* terhadap *use case* yang berelasi dengan *actor* tersebut. Gambar 2.14 menampilkan simbol relasi asosiasi pada *use case* sebagai berikut:

Gambar 2.14 Simbol relasi asosiasi di *use case diagram*

    b. Generalisasi adalah relasi antara *use case* umum (induk) dan *use case* lebih spesifik (anak). Relasi generalisai memungkinkan anak *use case* memiliki perilaku yang sama dengan induk *use case*. Relasi generalisasi digambarkan dalam bentuk anak panah segitiga. Gambar 2.15 menampilkan simbol relasi generalisasi pada *use case* sebagai berikut:

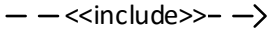


Gambar 2.15 Simbol relasi generalisasi di *use case diagram*

c. *Extend* adalah relasi yang memungkinkan terjadi penambahan beberapa perilaku (*behavior)* ke dalam *use case* awal yang pada dasarnya *use case* tersebut sudah dapat berdiri sendiri tanpa adanya penambahan. *Extend* dinotasikan dalam bentuk anak panah terputus dan terdapat tulisan <<*extend*>>> diatas panah tersebut. Gambar 2.16 menampilkan simbol relasi *extend* pada *use case* sebagai berikut:

Gambar 2.16 Simbol relasi *extend* di *use case diagram*

d. *Include* adalah relasi yang memungkinkan terjadi penambahan perilaku (*behavior)* ke dalam *use case* awal yang pada dasarnya *use case* ini tidak dapat berdiri sendiri tanpa adanya penambahan *use case* dan *use case* awal tidak akan lengkap tanpa adanya *use case* tambahan ini. *Use case* yang berada pada kepala anak panah adalah *use case* awal, dan pada sisi lain adalah *use case* penambah. *Include* dinotasikan dalam bentuk anak panah terputus dan terdapat tulisan <<*include*>>> diatas panah tersebut. Gambar 2.17 menampilkan simbol relasi *include* pada *use case* sebagai berikut:



Gambar 2.17 Simbol relasi *include* di *use case diagram*

2.12 *Activity Diagram*

Activity Diagram adalah mengambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. *Activity diagram* mengambarkan aktivitas sistem bukan apa yang dilakukan *actor*. Jadi aktivitas yang dapat dilakukan oleh sistem (Yulianto & Gartina, 2009). *Activity diagram* memiliki stuktur diagram yang mirip *flowchart* dan *data flow diagram* dalam perancangan terstruktur. Manfaat dari *activity diagram* adalah membantu dan memahami proses secara keseluruhan. *Activity* dibuat berdasarkan sebuah atau beberapa *use case* pada *use case diagram* (Fernandi, Alfandri, & Putri, 2013).

Ada beberapa elemen-elemen yang digunakan pada *activity diagram* yaitu *start state, activity, decision, join, end state, fork* dan *swimlane.* Berikut penjelasan dari elemen di *activity diagram* sebagai berikut (Yulianto & Gartina, 2009):

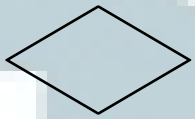
1. *Start state* adalah status awal dari aktivitas sistem. Gambar 2.18 menampilkan simbol dari *start state*.

Gambar 2.18 Simbol *start state* di *activity diagram*

2. *Activity* adalah aktivitas yang dilakukan pada sebuah sistem dan aktivitas biasanya diawalin dengan kata kerja. Gambar 2.19 menampilkan simbol dari *activity*.

Gambar 2.19 Simbol *activity* di *activity diagram*

3. Percabangan/*decision* adalah asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu. Gambar 2.20 menampilkan simbol dari *decision*.



Gambar 2.20 Simbol *decision* di *activity diagram*

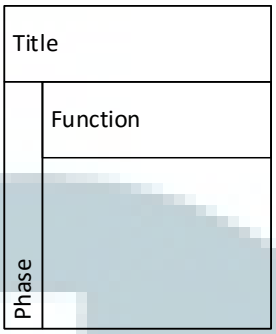
4. Pengabuangan/*join* adalah asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu. Gambar 2.21 menampilkan simbol dari *join*.

Gambar 2.21 Simbol *join* di *activity diagram*

5. *End state* adalah status akhir dari aktivitas sistem. Gambar 2.22 menampilkan simbol dari *end state*.

Gambar 2.22 Simbol *end state* di *activity diagram*

6. *Swinlane* adalah memisah aktor atau pengguna yang bertanggung jawab terhadap aktivitas yang terjadi. Gambar 2.23 menampilkan simbol dari *swinlane* vertikal.



Gambar 2.23 Simbol *swinlane* di *activity diagram*

7. *Fork* adalah kegiatan yang dilakukan secara paralel. Gambar 2.24 menampilkan simbol dari *fork*.

Gambar 2.24 Simbol *fork* di *activity diagram*