BAB II
TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

2.1.1 Logika *fuzzy*

Logika *fuzzy* atau *fuzzy logic* bermula dari kenyataan bahwa dunia nyata sangat kompleks. Kompleksitas ini muncul dari ketidakpastian dalam bentuk informasi *imprecision* (ketidakpastian). Mengapa komputer yang dibuat oleh manusia tidak mampu menangani persoalan yang kompleks sedangkan manusia bisa. Jawabannya adalah manusia mempunyai kemampuan untuk menalar (*reasoning*) dengan baik yaitu kemampuan yang komputer tidak miliki. Pada suatu sistem jika kompleksitasnya berkurang, maka persamaan matematis dapat digunakan dan ketelitian yang dihasilkan menjadi sangat berguna dalam pemodelan sistem tetapi jika kompleksitasnya bertambah dimana persamaan matematik tidak dapat digunakan, logika *fuzzy* menjadi salah satu alternatif penyelesaian. Logika *fuzzy* merupakan salah satu komponen pembentuk *soft computing*. Logika *fuzzy* pertama kali diperkenalkan oleh Prof. Lotfi A. Zadeh pada tahun 1965. Dasar logika *fuzzy* adalah teori himpunan *fuzzy* (Kusumadewi & Hari 2010), Logika *fuzzy* adalah suatu cara yang tepat untuk memetakan suatu ruang *input* kedalam suatu ruang *output* (Prabowo &Rahmadya 2012). Menurut (Cox 1994) terdapat beberapa alasan mengapa logika *fuzzy* digunakan adalah :

1. Konsep Logika *Fuzzy* mudah dipahami, karena logika *fuzzy* menggunakan dasar teori himpunan, maka konsep matematis yang mendasari penalaran *fuzzy* tersebut cukup mudah untuk dimengerti.
2. Logika *Fuzzy* sangat *fleksibel,* artinya mampu beradaptasi dengan perubahan-perubahan, dan ketidakpastian yang menyertai permasalahan.
3. Logika *Fuzzy* memiliki toleransi terhadap data yang tidak tepat, jika diberikan sekelompok data yang cukup *homogeny,* dan kemudian ada beberapa data yang "*eksklusif*", maka logika fuzzy memiliki kemampuan untuk menangani data *eksklusif* tersebut.
4. Logika Fuzzy mampu memodelkan fungsi-fungsi *nonlinier* yang sangat *kompleks.*

5. Logika *Fuzzy* dapat membangun dan mengaplikasikan pengalaman-pengalaman para pakar secara langsung tanpa harus melalui proses pelatihan. Dalam hal ini, sering dikenal dengan nama *Fuzzy Expert Systems* menjadi bagian terpenting.
6. Logika *Fuzzy* dapat diterapkan dalam desain sistem kontrol tanpa harus menghilangkan teknik desain sistem kontrol *konvensional* yang sudah ada terlebih dahulu. Hal ini umumnya terjadi pada aplikasi di bidang teknik mesin maupun teknik elektro.
7. Logika *Fuzzy* didasarkan pada bahasa alami

Logika *Fuzzy* adalah sebuah logika yang memiliki nilai kekaburan atau kesamaran antara benar dan salah. Dalam teori Logika *Fuzzy*, sebuah nilai bisa bernilai benar dan salah secara bersamaan namun berapa besar kebenaran dan kesalahan suatu nilai tergantung kepada bobot keanggotaan yang dimiliki. Logika *Fuzzy* memiliki nilai antara 0,0 sampai 0,1. Nilai ini adalah nilai keanggotaan atau derajat keanggotaan.

Logika *Fuzzy* merupakan pengetahuan yang memiliki kemampuan untuk menjembatani bahasa mesin yang serba terukur dengan bahasa manusia yang cenderung tidak terukur. Misalnya saja bahasa manusia untuk suhu udara (panas, sedang, dingin), panjang badan seseorang (tinggi, sedang, pendek), umur manusia (muda, tua, paruh baya), sampai pada penilaian terhadap pekerjaan (baik, sedang, jelek). Masing-masing memiliki penilaian tersendiri tentang hal-hal di atas. Penilaian baik, sedang, jelek, panas, dingin, tinggi, pendek, muda, tua, dll; tidak bisa diidentifikasi dengan jelas. Dengan Logika *Fuzzy*, bahasa manusia diimplementasikan ke dalam bahasa mesin secara mudah dan efesien.

2.1.2 Himpunan *fuzzy*

Pada himpunan tegas (*crisp*), nilai keanggotaan suatu item di dalam sebuah himpunan memiliki 2 kemungkinan, yaitu:

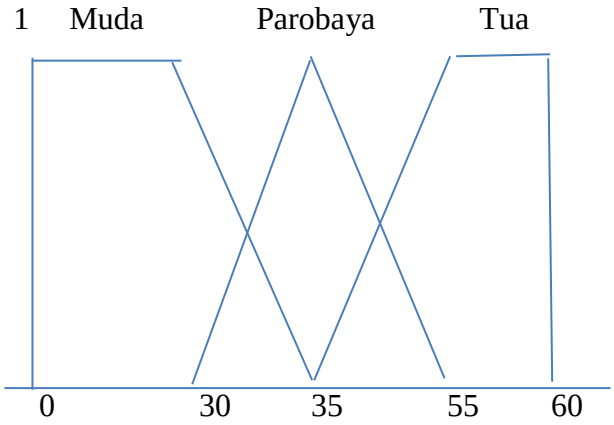
1. 1 (satu) yang berarti *item* tersebut merupakan anggota dalam himpunan
2. 0 (nol) yang berarti *item* tersebut bukan merupakan anggota dalam himpunan

Misal, variabel umur terbagi menjadi 3 himpunan *fuzzy*, yaitu :

1.Muda ; umur seseorang < 35 tahun

2.Parobaya; 30 tahun $\leq$ umur seseorang ≤ 55 tahun

3.Tua; umur seseorang > 35 tahun



Gambar 2.1 Himpunan *fuzzy* untuk umur

Dari pembagian anggota himpunan diatas, dapat disimpulkan:

a. Apabila seseorang dengan umur 25 tahun termasuk dalam kategori Muda.
b. Apabila seseorang dengan umur 40 tahun termasuk dalam kategori Parobaya.
c. Apabila seseorang dengan umur 56 tahun termasuk dalam kategori Tua.

Dapat kita lihat bahwa penggunaan himpunan *crisp* untuk menyatakan umur seseorang tidaklah 'adil'. Perubahan kecil pada sebuah nilai akan mengakibatkan perbedaan kategori yang cukup signifikan.

'Ketidakadilan' ini dapat diatasi dengan himpunan *fuzzy*.Dalam himpunan *fuzzy*, nilai keanggotaan terletak di antara 0 dan 1.Ini berarti jika *item* bernilai 0, maka *item* tersebut bukan merupakan anggota dari sebuah himpunan, namun bila *item* memiliki nilai keanggotaan 1, maka item tersebut mempunyai keanggotaan penuh pada himpunan itu.

Himpunan *fuzzy* memiliki 2 atribut :

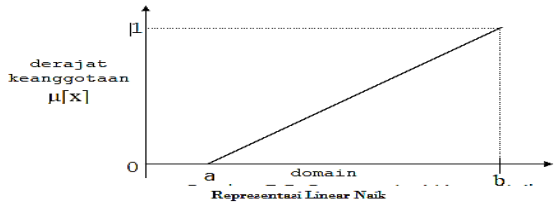
1. *Linguistic*, yaitu penamaan suatu grup yang mewakili suatu keadaan/kondisi tertentu dengan menggunakan bahasa manusia. Misalnya: tinggi, pendek, panas, dingin, cepat, lambat, sedang.
2. *Numeris*, yaitu nilai yang menunjukan ukuran dari sebuah variabel. Misalnya : 4, 23, 40, 46.

2.1.3 Fungsi keanggotaan

Fungsi keanggotaan *(membership function)* adalah suatu kurva yang menunjukan dan mendefinisikan bagaimana tiap titik dalam ruang input dipetakan menjadi bobot atau derajat keanggotaan antara 0 dan 1. Salah satu cara yang dapat digunakan untuk mendapatkan nilai keanggotaan adalah melalui pendekatan fungsi. Ada beberapa fungsi yang dapat digunakan:

1. *Representasi Linier*

Pada *representasi linier*, pemetaan input derajat keanggotaannya digambarkan sebagai suatu garis lurus. Bentuk ini sangat sederhana dan pilihan yang baik untuk konsep yang kurang jelas. Ada 2 keadaan himpunan *fuzzy linier*, yaitu : *linier* naik dan *linier* turun.Untuk representasi linier naik, kenaikan himpunan dimulai dari nilai domain yang memiliki derajat keanggotaan 0 bergerak ke kanan menuju nilai *domain* yang memiliki derajat keanggotaan lebih tinggi dan dapat digambarkan seperti gambar 2.2 berikut.



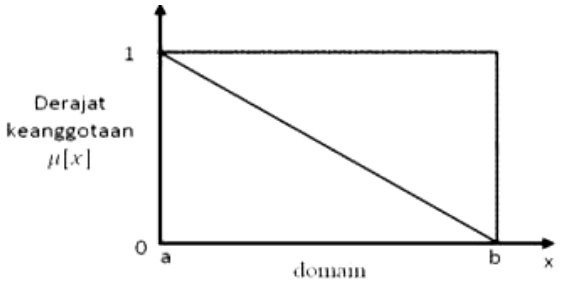
Gambar 2.2 *Representasi Linier* Naik

Fungsi keanggotaan dari *representasi linier* naik ini dapat disimpulkan sebagai berikut:

$$\mu\,[x] \quad = \begin{cases} 0 & ; \quad x \le a \\ (x-a)/(b-a) & ; \quad a \le x \le b \\ 1 & ; \quad x \ge b \end{cases}$$

Untuk *representasi linier* turun, garis lurus dimulai dari nilai *domain* dengan derajat keanggotaan tertinggi (1) dan bergerak ke kanan menuju ke nilai domain dengan derajat keanggotaan lebih rendah.

Untuk gambar *representasi linier* turun, dapat dilihat pada gambar 2.3 dibawah ini:



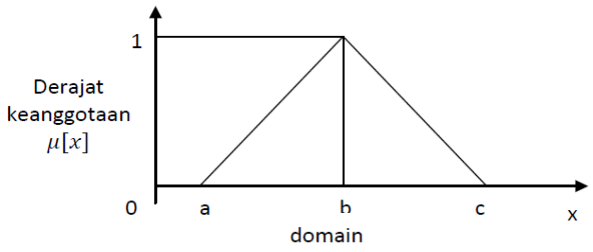
Gambar 2.3*Representasi Linier* Turun (Kusumadewi 2013)

Dan fungsi keanggotaan dari *representasi linier* turun dapat dituliskan sebagai berikut:

$$\mu\,[x] \quad = \begin{cases} (b - x)/(b - a) & ;\ a \le x \le b \\ 0 & ;\ x \ge b \end{cases}$$

2. *Representasi* Kurva Segitiga

Kurva segitiga pada dasarnya merupakan gabungan dari 2 garis *linier* (garis *linier* naik dan *linier* turun) yang dapat dilihat pada gambar 2.4:



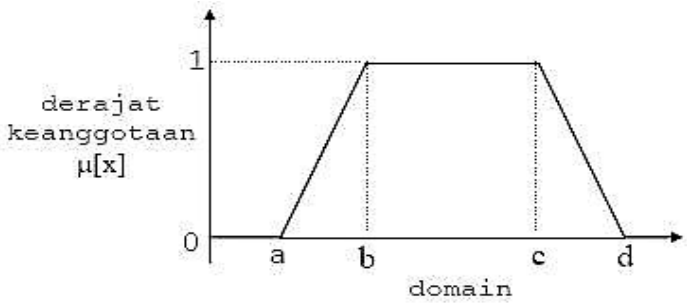
Gambar 2.4 *Representasi* Kurva Segitiga (Kusumadewi 2013)

Dengan fungsi keanggotaan sebagai berikut:

$$\mu\,[x] \quad = \begin{cases} 0 & ;\ x \le a \text{ atau } x \ge c \\ (x - a)/(b - a) & ;\ a \le x \le b \\ (b - x)/(c - b) & ;\ b \le x \le c \end{cases}$$

3. *Representasi* Kurva Trapesium

Memiliki bentuk dasar yang hampir mirip dengan bentuk kurva segitiga, hanya saja ada beberapa nilai yang bernilai keanggotaan 1. Dapat dilihat pada gambar 2.5:



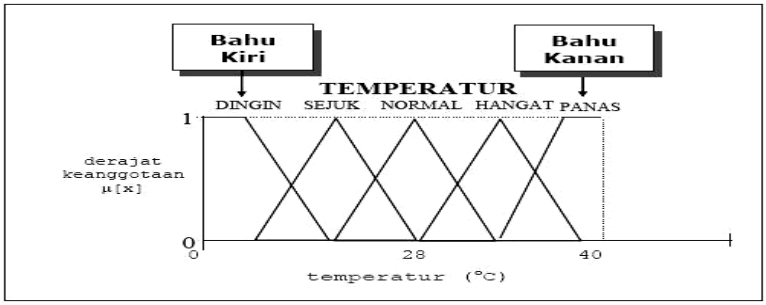
Gambar 2.5 *Representasi* Kurva Trapesium (Kusumadewi 2013)

Dengan fungsi keanggotaan sebagai berikut :

$$\mu[x] = \begin{cases} 0; & x \le a \text{ atau } x \ge d \\ (x-a)/(b-a); & a \le x \le b \\ 1; & b \le x \le c \\ (d-x)/(d-c); & x \ge d \end{cases}$$

4. *Respresentasi* Kurva Bentuk Bahu

Dalam *representasi* kurva bentuk bahu, daerah yang terletak di tengah-tengah suatu variabel yang dipresentasikan dalam bentuk segitiga, pada sisi kanan dan kirinya akan naik dan turun. Hal inilah yang dimaksudkan bahwa suatu nilai dapat menjadi anggota dari 2 himpunan yang berbeda, bergantung pada derajat keanggotaan nilai tersebut dalam himpunan (misalkan : DINGIN bergerak ke SEJUK bergerak ke HANGAT dan bergerak ke PANAS). Sebagai contoh, apabila telah mencapai kondisi PANAS, kenaikan temperatur akan tetap berada pada kondisi PANAS. 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. Gambar II.9 dibawah ini adalah gambar kurva bentuk bahu pada kasus temperatur udara. Dapat dilihat bahwa suhu normal di 28˚C dan panas di atas 40˚C, serta dingin di bawah 15˚C.Di antara 28˚C dan 40˚C adalah hangat, dan di antara 28˚C dan 15˚C adalah sejuk.

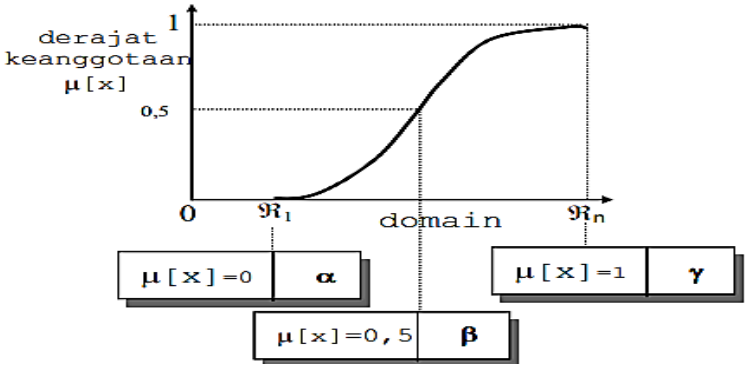


Gambar 2.6 *Representasi* Kurva Bentuk Bahu (Kusumadewi 2013)

5. *Representasi* Kurva S

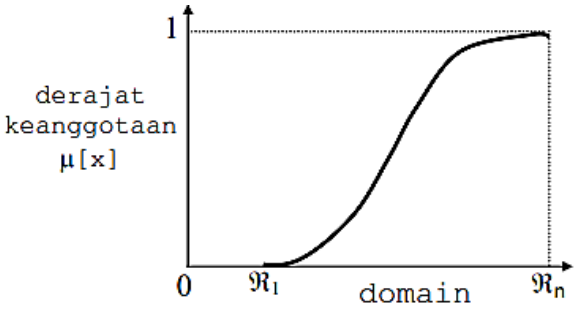
Sama seperti *representasi* linier, representasi kurva S juga memiliki 2 jenis, yaitu kurva pertumbuhan dan kurva penyusutan. Bedanya, pada kurva S, kenaikan dan penurunan permukaan secara tak linier.Kurva S pertumbuhan akan bergerak dari sisi paling kiri dimana nilai keanggotaanya 0 ke sisi kanan yang nilai anggotanya 1.Dalam representasi kurva S ada 3 parameter utama, yaitu nilai keanggotaan nol (α), nilai keanggotaan lengkap (γ) dan titik *infleksi* atau sering disebut *crossover* (β) yaitu titik yang memiliki nilai fungsi keanggotaanya 50% atau 0,5(Cox 1994).

Karakteristik kurva S dapat dilihat pada gambar 2.7 di bawah ini:



Gambar 2.7 *Representasi* Fungsi Kurva S (Cox 1994)

Sementara untuk kurva S pertumbuhan dapat dilihat pada gambar 2.8.sebagai berikut:



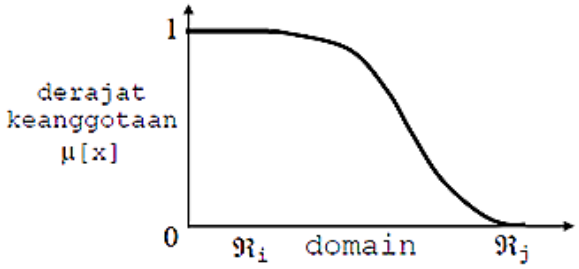
Gambar 2.8 *Representasi* Fungsi Kurva S Pertumbuhan (Cox 1994)

Dengan fungsi keanggotaan sebagai berikut :

$$S(x;\alpha,\beta,\gamma)=\begin{cases} 0 & \rightarrow \quad x \le \alpha \\ 2((x-\alpha)/(\gamma-\alpha))^2 & \rightarrow \alpha \le x \le \beta \end{cases}$$

$$1\text{-}2((\gamma - x)/(\gamma - \alpha))^2 \quad \rightarrow \beta \leq x \leq \alpha$$
$$1 \quad \rightarrow \quad x \geq \gamma$$

Sedangkan untuk kurva S penyusutan dapat dilihat pada gambar 2.9. sebagai berikut:



Gambar 2.9 *Representasi* Fungsi Kurva S Penyusutan (Cox 1994)

Dengan fungsi keanggotaan sebagai berikut :

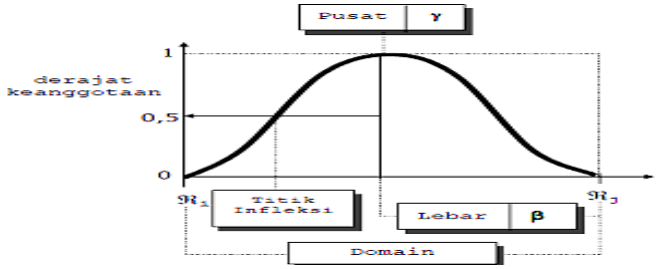
$$S(x;\alpha,\beta,\gamma) = \begin{cases} 0 & \rightarrow \quad x \leq \alpha \\ 1\text{-}2((x - \alpha)/(\gamma - \alpha))^2 & \rightarrow \alpha \leq x \leq \beta \\ 2((\gamma - x)/(\gamma - \alpha))^2 & \rightarrow \beta \leq x \leq \alpha \\ 1 & \rightarrow \quad x \geq \gamma \end{cases}$$

6. *Representasi* Kurva Bentuk Lonceng

Terbagi atas 3 jenis menurut perbedaan gradienya, yaitu:

a. Himpunan *fuzzy* Pi

Kurva Pi berbentuk lonceng dengan derajat keanggotaan 1 terletak pada pusat dengan domain γ dan lebar kurva β. Dapat dilihat pada gambar 2.10:



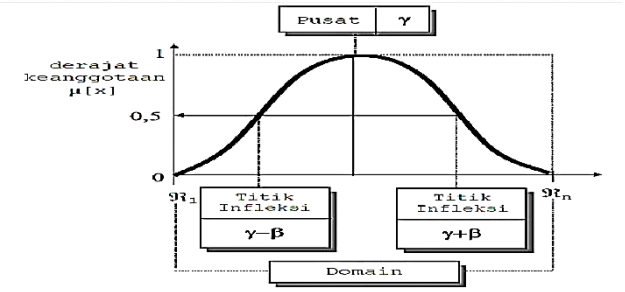
Gambar 2.10 *Representasi* Fungsi Kurva Lonceng Pi (Cox 1994)

Dengan fungsi keanggotaan sebagai berikut :

$$\pi\,(x,\beta,\gamma) = \begin{cases} s\ (x;\ \gamma-\beta,\gamma-\frac{\beta}{2},\gamma) & ;\ x \le \gamma \\ 1-s\left(x;\ \gamma,\gamma+\frac{\beta}{2},\gamma+\beta\right) & ;\ x > \gamma \end{cases}$$

b. Beta

Kurva ini didefinisikan dengan nilai pada domain yang menunjukan pusat kurva γ dan setengah lebar kurva β. Dapat dilihat pada gambar 2.11:



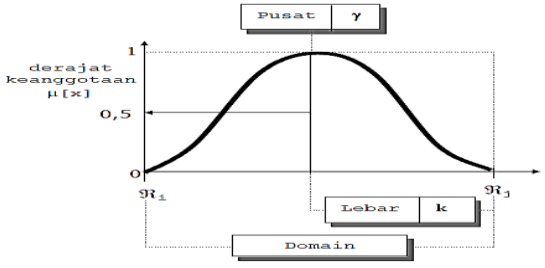
Gambar 2.11 *Representasi* Fungsi Kurva Lonceng Beta (Cox 1994)

Dengan fungsi keanggotaan sebagai berikut :

$$B\ (x;\ \gamma,\beta) = \frac{1}{1+\left(\frac{x-\gamma}{\beta}\right)^2}$$

c. *Gauss*

Pada kurva *Gauss*, nilai domain pada pusat kurva dilambangkan dengan γ dah k menunjukan lebar kurva. Dapat dilihat pada gambar 2.12 :



Gambar 2.12 *Representasi* Fungsi Kurva Lonceng Gauss (Cox 1994)

Dengan fungsi keanggotaan sebagai berikut :

$$G(x;k,y) = e^{-k(y-x)^2}$$

2.1.3.1 Operator dasar logika *fuzzy*

Ada beberapa operasi yang didefinisikan untuk memodifikasi atau mengkobinasi himpunan *fuzzy*. Nilai keanggotaan dari hasil operasi 2 himpunan sering disebut dengan *free strength (α*-predikat). Ada 3 operator dasar yang diciptakan oleh Zadeh sebagai operator dasar dalam logika Fuzzy (Cox 1994), yaitu:

1. Operator AND

   Digunakan untuk interaksi pada himpunan.α-predikat dari dua himpunan yang dimodifikasi dengan operator AND adalah nilai keanggotaan terkecil antar elemen pada himpunan-himpunan yang bersangkutan.

$$\mu_{A\cap B}=\min(\mu_{A(X)}, \mu_{B(y)})$$

2. Operator OR

   Berhubungan dengan operasi union pada himpunan.α-predikat dari dua himpunan yang dimodifikasi dengan operator OR adalah nilai keanggotaan terbesar anatar elemen pada himpunan-himpunan yang bersangkutan.

$$\mu_{A\cap B}=\max(\mu_{A(X)}, \mu_{B(y)})$$

3. Operator NOT

   Berhubungan dengan operasi komplemen pada himpunan.α-predikat dari himpunan yang dimodifikasi dengan operator NOT adalah nilai keanggotaan himpunan tersebut dikurang 1.

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

2.1.3.2 IF-THEN rule

Fuzzy logic bekerja dengan aturan-aturan yang dinyatakan dalam bentuk IF-THEN. Sebuah aturan *fuzzy* tunggal berbentuk seperti dibawah ini :

If x is A then y is B

A dan B adalah *linguistic values* (seperti panas, dingin, tinggi, pendek, besar, kecil, baik, buruk,dll) yang didefinisikan diantara rentang variabel x dan y. Pernyataan "x is A" disebut *antecedent* (premis*)* sementara pernyataan "y is B" disebut *consequent* (kesimpulan).

Mengintepretasikan sebuah IF-THEN *rules* meliputi 2 bagian, yaitu:

1. Mengevaluasi *antecedent*, yaitu melakukan *fuzzifikasi* pada input dan menerapkan operasi-operasi *fuzzy logic* dengan operator-operator *fuzzy*.
2. Proses implikasi, yaitu menerapkan hasil operasi *fuzzy logic* pada bagian *antecedent* untuk mengambil kesimpulan. Dengan menggunakan IF-THEN *rule* tunggal, sebenarnya tidaklah cukup untuk mendapatkan keputusan terbaik.

2.1.3.3 Fungsi implikasi

Secara umum, ada 2 fungsi implikasi yang dapat digunakan :

1. *Min (*Minimum)

Fungsi ini akan memotong output himpunan

2. *Dot* (Produk)

Fungsi ini akan menskala output himpunan fuzzy

2.2 Sistem Pendukung Keputusan

Sistem Pendukung Keputusan atau *Decision Support System* (DSS) secara umum didefinisikan sebagai sebuah sistem yang mampu memberikan kemampuan pemecahan masalah maupun kemampuan pengomunikasian untuk masalah semi terstruktur (Turban 2005).

Sistem Pendukung Keputusan (SPK) adalah bagian dari sistem informasi berbasis computer termasuk system berbasis pengetahuan untuk mendukung pengambilan keputusan dalam suatu organisasi maupun perusahaan (Asfi, 2010: 2).

SPK adalah sebuah sistem informasi yang berbasis komputer yang mampu memanfaatkan data dan model untuk menyelesaikan masalah-masalah dan memberi solusi alternatif sehingga memudahkan pengambilan keputusan terhadap suatu masalah.

Menurut Turban(2005) menjelaskan terdapat sejumlah karakteristik dan kemampuan SPK yaitu:

1. SPK merupakan sistem berbasis komputer dengan antarmuka antara mesin/komputer dengan pembuat keputusan.
2. Memberikan hak penuh kepada pembuat keputusan untuk mengontrol seluruh tahap dalam proses pembuatan keputusan.
3. SPK mampu memberi solusi bagi masalah tidak terstruktur, baik bagi perorangan atau kelompok.
4. SPK mengunakan data, basis data, dan analitis metode-metode keputusan.
5. Kemampuan SPK adalah dapat melakukan adaptasi setiap saat dan bersifat fleksibel.
6. SPK ditujukan untuk membantu pembuat keputusan dalam menyelasaikan masalah dan bukan menggantiposisimanusia sebagai pembuat keputusan.

Menurut Herbert A. Simon (Asfi, 2010: 2) proses pengambilan keputusan mempunyai 3 tahap,yaitu:

1. Pemahaman

   Menyelidiki lingkungan kondisi-kondisi yang memerlukan keputusan data mentah yang diperoleh, diolah dan diperiksa untuk dijadikan petunjuk yang dapat menentukan masalahnya.

2. Perancangan

   Menemukan, mengembangkan, dan menganalisis arah tindakan yang mungkin dapat dipergunakan. Hal ini mengandung proses-proses untuk memahami masalah, untuk menghasilkan cara pemecahan, dan untuk menguji apakah cara pemecahan tersebut dapat dilaksanakan.

3. Pemilihan

   Memilih arah tindakan tertentu dari semua arah tindakan yang ada. Pilihan ditentukan dan dilaksanakan.

2.3 *Analitical Hierarchy Process* (AHP)

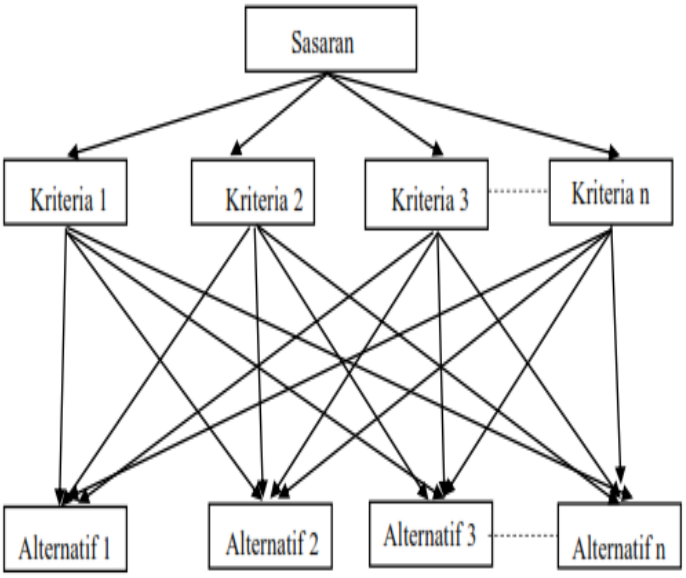
AHP merupakan suatu metode pendekatan yang sesuai untuk menangani sistem kompleks yang berhubungan dengan penentuan keputusan dari beberapa alternatif dan memberikan pilihan yang dapat dipertimbangkan. Metode ini dikembangkan pertama kali oleh Saaty (Saaty,1980).

Model hierarki yang dinyatakan oleh Saaty adalah model hierarki fungsional dengan input utamanya adalah persepsi manusia.

Dalam menyelesaikan permasalahan dengan AHP ada beberapa prinsip yang harus dipahami, diantaranya adalah sebagai berikut:

1. Dekomposisi(*Decomposition*)

   Sistem yang kompleks dapat dipahami dengan memecahkannya menjadi elemen-elemen yang lebih kecil dan sehingga mudah dipahami. Kemudian disusun secara hieraki seperti Gambar 2.13



Gambar 2.13 Model AHP

2. Penilaian Komparatif (*Comparative judgment*)

   Kriteria dan alternatif dilakukan dengan perbandingan berpasangan. Menurut Saaty (2008:86), untuk berbagai persoalan, skala 1 sampai 9 adalah skala terbaik untuk mengekspresikan pendapat. Nilai dan definisi pendapat kualitatif dari skala perbandingan Saaty dapat diukur menggunakan tabel analisis seperti Tabel 2.1

Tabel 2.1 Skala Penilaian Perbandingan Berpasangan

Intensitas Kepentingan	Keterangan
1	Kedua elemen sama pentingnya

3	Elemen yang satu sedikit lebih penting daripada elemen yang lainnya
5	Elemen yang satu lebih penting daripada yang lainnya
7	Satu elemen sangat kuat penting daripada elemen lainnya
9	Satu elemen amat sangat penting dari pada elemen lainnya
2,4,6,8	Apabila ragu-ragu antara dua nilai yang saling berdekatan

3. Sistesis Prioritas (*Synthesis of priority*)

   Menentukan prioritas dari elemen-elemen kriteria dapat dipandang sebagai bobot/kontribusi elemen tersebut terhadap tujuan pengambilan keputusan.AHP , melakukan analisis prioritas elemen dengan metode perbandingan berpasangan antar dua elemen sehingga semua elemen yang ada tercakup. Prioritas ini ditentukan berdasarkan pandangan para pakar dan pihak-pihak yang berkepentingan terhadap pengambilan keputusan, baik secara langsung(diskusi) maupun secara tidak langsung (*kuesioner*).

4. Konsistensi Logis (*Logical Consistency*)

   Konsistensi memiliki dua makna. Pertama, objek-objek yang serupa bisa dikelompokkan sesuai dengan keseragaman dan relevansi. Kedua, menyangkut tingkat hubungan antarobjek yang didasarkan pada kriteria tertentu (Kosasi,2002: 89).

Secara umum langkah-langkah dalam menggunakan metode AHP untuk pemecahan suatu masalah adalah sebagai berikut (Manurung,2010:30-32):

1. Mendefinisikan masalah dan menentukan solusi yang diinginkan, lalu menyusun hierarki dari permasalahan yang dihadapi.
2. Menentukan prioritas elemen
   a. Langkah pertama dalam menentukan prioritas elemen adalah membuat perbandingan pasangan, yaitu membandingkan elemen secara berpasangan sesuai kriteria yang diberikan.

b. Matriks perbandingan berpasangan diisi menggunakan bilangan untuk merepresentasikan kepentingan relatif dari suatu elemen terhadap elemen yang lainnya. Matriks K merupakan matriks perbadingan berpasangan antarkriteria.

$$K = \begin{array}{c} \\ K_1 \\ K_2 \\ \vdots \\ K_m \end{array} \begin{array}{c} \begin{array}{cccc} K_1 & K_2 & \cdots & K_n \end{array} \\ \begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1n} \\ K_{21} & K_{22} & \cdots & K_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ K_{m1} & K_{m2} & \cdots & K_{mn} \end{bmatrix} \end{array}$$

c. Mengubah matriks perbandingan berpasangan menjadi matriks skala perbandingan tingkat kepentingan *fuzzy*

Tabel 2.2 Matriks berpasangan tingkat kepentingan *fuzzy*

Bobot	Tingkat Skala Fuzzy			Invers Skala Fuzzy			Definisi Variabel Linguistik
	L	M	U	L	M	U	
1	1	1	1	1	1	1	Sama Penting
2	1	1	1,5	0,667	1	1	Ragu-ragu
3	1	1,5	2	0,5	0,66667	1	Sedikit Lebih Penting
4	1,5	2	2,5	0,4	0,5	0,667	Ragu-ragu
5	2	2,5	3	0,333	0,4	0,5	Lebih Penting
6	2,5	3	3,5	0,286	0,33333	0,4	Ragu-ragu
7	3	3,5	4	0,25	0,28571	0,333	Sangat Kuat Penting
8	3,5	4	4,5	0,222	0,25	0,286	Ragu-ragu
9	4	4,5	4,5	0,222	0,22222	0,25	Amat Sangat Penting

3. Sintesis

Pertimbangan-pertimbangan terhadap perbandingan berpasangan di sintesis untuk memperoleh keseluruhan prioritas. Hal-hal yang dilakukan dalam langkah ini adalah:

a. Menjumlahkan nilai-nilai dari setiap kolom pada matriks K.
b. Membagi setiap nilai dari kolom dengan total kolom yang bersangkutan untuk memperoleh normalisasi matriks.
c. Menjumlahkan nilai-nilai dari setiap baris dan membaginya dengan jumlah elemen untuk mendapatkan nilai bobot prioritas.

4. Mengukur Konsistensi

Dalam pembuatan keputusan, penting untuk mengetahui seberapa baik konsistensi yang ada karena kita tidak menginginkan keputusan berdasarkan pertimbangan dengan konsistensi yang rendah. Hal-hal yang dilakukan dalam langkah ini adalah sebagai berikut:

a. Setiap nilai pada kolom pertama dikalikan dengan bobot prioritas elemen pertama, kemudian setiap nilai pada kolom kedua dikalikan dengan bobot prioritas elemen kedua dan seterusnya.
b. Jumlahkan setiap baris (∑ baris).
c. Hasil dari penjumlahan baris dibagi dengan elemen prioritas yang bersangkutan sehingga didapat lamda.

$$\lambda = \frac{\Sigma baris}{prioritas}$$

d. Jumlahkan lamda (λ) dan hasilnya dibagi dengan banyaknya elemen yang ada, hasilnya disebut λ maks.

$$\lambda_{max} = \frac{\Sigma \lambda}{n}$$

dengan n = banyaknya elemen yang dibandingkan

e. Hitung Indeks Konsistensi/*Consistency Index* (CI) dengan rumus:

Ukuran Matriks	Nilai RC

$$CI = \frac{(\lambda \text{max} - \text{n})}{n - 1}$$

dengan n = banyaknya elemen yang dibandingkan

f. Hitung Rasio Konsistensi/*Consistency Ratio* (CR) dengan rumus:

$$CR = \frac{CI}{RC}$$

dengan:

CR = *Consistency Ratio/*konsistensi rasio

CI = *Consistency Index/*indeks konsistensi

RC = *Random Consistency*/konsistensi random

g. Nilai RC sudah ditentukan berdasarkan matriks perbandingan yang dibentuk dan dapat disajikan pada tabel 2.3

Tabel 2.3 Nilai *Random Consistensy* (RC)

1,2	0,00
3	0,58
4	0,90
5	1,12
6	1,24
7	1,32
8	1,41
9	1,45
10	1,49
11	1,51
12	1,53
13	1,56
14	1,57

Sumber: Saaty (1994)

h. Memeriksa konsistensi hierarki

Jika nilainya lebih dari 10%, maka penilaian data judgement harus diperbaiki. Namun jika Rasio Konsistensi $(\frac{CI}{RC})$ kurang atau sama dengan 0,1 maka hasil perhitungan dinyatakan benar (Kusrini, 2007).

2.4 *Technique For Order Preference by Similarity to Ideal Solution* (TOPSIS)

Technique For Order Preference by Similarity to Ideal Solution atau TOPSIS merupakan salah satu metode pengambilan keputusan yang pertama kali diperkenalkan oleh Yonn dan Hwang (1981). Ide dasar dari metode ini adalah alternatif yang dipilih memiliki jarak terdekat dengan solusi ideal positif dan memiliki jarak terjauh dari solusi ideal negatif. TOPSIS memperhatikan jarak ke solusi ideal positif maupun solusi ideal negatif dengan mengambil hubungan kedekatan menuju solusi ideal. Dengan melakukan perbandingan pada keduanya, urutan pilihan dapat ditentukan.

Dalam metode TOPSIS secara garis besar terdapat 6 langkah sebagai berikut (Mahmoodzaadeh, 2007: 305):

a. Konversi dan Bentuk Matriks Keputusan

Setelah masing-masing kriteria pada alternatif diberi nilai fuzzy, kemudian dihitung nilai total integeral untuk setiap alternatif menggunakan persamaan (1).

$$x = I(F) = \frac{1}{2}\left((1-\alpha)A + B + \alpha C\right) \ldots\ldots\ldots\ldots\ldots (1)$$

Dengan α adalah h derajat tingkat keoptimisan dengan nilai antara 0 sampai 1. Dalam penelitian ini nilai $\alpha = 0{,}5$

Bentuk matrik keputusan sebagai berikut

$$D = \begin{array}{c} \\ A_1 \\ A_2 \\ \vdots \\ A_m \end{array} \begin{array}{c} \begin{array}{cccc} F_1 & F_2 & \cdots & F_n \end{array} \\ \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1n} \\ f_{21} & f_{22} & \cdots & f_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ f_{m1} & f_{m2} & \cdots & f_{mn} \end{bmatrix} \end{array}$$

dengan A_i adalah alternatif dengan i = 1, 2, ..., m. F_j adalah atribut atau kriteria dengan j = 1, 2, ... , n. Sedangkan F_{ij} adalah alternatif ke - i dan kriteria ke-j.

b. Normalisasi Matriks Keputusan

Setiap elemen pada matriks D dinormalisasikan untuk mendapatkan matriks normalisasi R. Normalisasi nilai r_{ij} adalah sebagai berikut:

$$r_{ij} = \frac{f_{ij}}{\sqrt{\sum_{i=1}^{m} f_{ij}^2}} \quad \ldots\ldots\ldots (2)$$

dengan

i = 1, … m,

j = 1, ... n,

c. Pembobotan Normalisasi

Menghitung besarnya bobot pada matriks keputusan yang telah dinormalisasi, didapat dari mengkalikan hasil normalisasi matriks keputusan dengan bobot kriteria. Matriks V_{ij} dari Pembobotan Normalisasi diperoleh dari:

$$vij = wij * rij \quad \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

dengan:

w_j adalah bobot kriteria dari matriks bobot ($W = w1, w2, \dots, wn$). Sehingga didapat matriks sebagai berikut:

$$V = \begin{bmatrix} V_{11} & V_{12} & \cdots & V_{1n} \\ V_{21} & V_{22} & \cdots & V_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ V_{m1} & V_{m2} & \cdots & V_{mn} \end{bmatrix}$$

d. Solusi Ideal Positif dan Negatif

Solusi ideal positif dinotasikan sebagai A_+ dan solusi ideal negatif dinotasikan dengan A_-, untuk menentukan sulosi ideal positif dan negatif menggunakan cara sebagai berikut:

$$A^+ = \{(max v_{ij} | j \in J), (min v_{ij} | j \in J'), i = 1,2,3, \dots m\}$$
$$= \{v_1^+, v_2^+, \dots v_m^+\}$$
$$A^- = \{(min v_{ij} | j \in J), (max v_{ij} | j \in J'), i = 1,2,3, \dots m\}$$
$$= \{v_1^-, v_2^-, \dots v_m^-\}$$

dengan

v_{ij} = elemen matriks V baris ke-i dan kolom ke-j

J = {j=1,2,3,...,n dan j berhubungan dengan *benefit criteria*}

J' = {j=1,2,3,...,n dan j berhubungan dengan *cost criteria*}

e. *Separation Measure*

Separation Measure adalah pengukuran jarak dari suatu alternatif ke solusi ideal positif dan solusi ideal negatif. Perhitungannya sebagai berikut:

Separation measure untuk solusi ideal positif

$$D_i^+ = \sqrt{\sum_{j=1}^{n} (v_{ij} - v_j^{\cdot}} \quad \text{.....(4)}$$

dengan i = 1, ..., m

Separation measure untuk solusi ideal negatif

.....(5)

$$D_i^- = \sqrt{\sum_{j=1}^{n} \left(v_{ij} - v_j^-\right)^2}$$

dengan i = 1, ..., m

f. Kedekatan Relatif

Kedekatan relatif dari alternatif solusi ideal positif Adengan solusi ideal negatif A- direpresentasikan sebagai berikut:

$$C_i = \frac{L}{D_i^+} \quad \text{.....(6)}$$

dengan $0 < C_i < 1$ dan i=1,2,3,..., m

g. Mengurutkan Pilihan

Hasil akhir adalah pengurutan alternatif yang dirangking berdasarkan urutan C_i. Sehingga solusi alternatif terbaik adalah salah satu yang berjarak terpendek dari solusi ideal positif dan berjarak terjauh dari solusi ideal negatif.

2.5 Promosi Jabatan

Menurut Manullang (2001: 153) "Promosi berarti kenaikan jabatan yakni menerima kekuasaan dan tanggung jawab lebih besar dari kekuasaan dan tanggung jawab sebelumnya". Sedangkan menurut Hasibuan (2002: 108) "Promosi adalah perpindahan yang memperbesar *authorithy* dan *responsibility* karyawan ke jabatan yang lebih tinggi di dalam suatu organisasi sehingga kewajiban, hak, status dan penghasilan lebih besar".

Menurut Hasibuan (2002: 113) ada beberapa tujuan dilaksanakannya promosi jabatan yaitu :

1. Untuk memberikan pengakuan, jabatan dan imbalan jasa yang semakin besar kepada karyawan yang berprestasi kerja tinggi.
2. Dapat meinimbulkan kepuasan dan kebanggaan pribadi, status sosial yang semakin tinggi dan penghasilan yang semakin besar.

3. Untuk merangsang agar karyawan lebih bergairah bekerja, berdisiplin tinggi dan memperbesar kinerjanya.
4. Untuk menjamin stabilitas kepegawaian.
5. Memperbaiki status karyawan dari karyawan sementara menjadi karyawan tetap setelah lulus dalam masa percobaan.

2.6 Perangkat Lunak Aplikasi

2.6.1 PHP

PHP adalah singkatan dari *Personal Hypertext Preprocessor*. Ia merupakan bahasa berbentuk *scripting* yang menyatu dalam HTML dan dijalankan pada *server side*. Artinya semua sintaks yang kita berikan sepenuhnya dijalankan pada *server* sedangkan yang dikirimkan ke *browser* hanya hasilnya saja. (Arhami, 2005).

Kelebihan PHP yaitu:

1. PHP mudah dibuat dan kecepatan akses tinggi.
2. PHP dapat berjalan dalam *web server* yang berbeda dan dalam sistem operasi yang berbeda pula. PHP dapat berjalan di sistem operasi UNIX, Windows98, Windows NT, dan Manchitosh.
3. PHP diterbitkan secara gratis.
4. PHP juga dapat berjalan pada *web server* Microsoft Personal Web Server, Apache, IIS, Sitami, dan sebagainya.
5. PHP termasuk bahasa yang *embedded* (bisa ditempel atau diletakkan dalam *tag* HTML).
6. PHP termasuk *server side programming*. Program PHP membagi tipe data menjadi lima jenis data, yaitu *integer, floating point, string*, dan *array* (Sidik, 2006: 38).

PHP pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1995 dan bersifat *open source* yang ditulis menggunakan sintaks bahasa C, Java, dan Perl. Pada waktu itu PHP masih bernama FI (*Form Interpreted*) yang wujudnya berupa sekumpulan *script* yang digunakan untuk mengolah data *form* dari *web.* Pada tahun 1997 sebuah perusahaan bernama Zend menulis ulang *interpreter* PHP menjadi lebih bersih, lebih baik, dan lebih cepat. Kemudian, pada Juni 1998 perusahaan tersebut merilis *interpreter* baru untuk PHP dan meresmikan rilis tersebut sebagai

PHP 3.0. Pada pertengahan tahun 1999, Zend merilis *interpreter* PHP baru dan rilis tersebut dikenal dengan PHP 4.0. PHP 4.0 adalah versi PHP yang paling banyak dipakai pada awal abad ke-21. Versi ini banyak dipakai disebabkan kemampuannya untuk membangun aplikasi *web* kompleks tetapi tetap memiliki kecepatan dan stabilitas yang tinggi. Pada Juni 2004, Zend merilis PHP 5.0. Dalam versi ini, inti dari *interpreter* PHP mengalami perubahan besar. Versi ini juga memasukkan model pemrograman berorientasi objek ke dalam PHP untuk menjawab perkembangan bahasa pemrograman ke arah paradigma berorientasi objek. PHP hampir dapat berjalan di semua sistem operasi seperti Windows, Unix, Linux, dan variannya, Mac OS X, RISC OS dan sistem operasi lainnya. PHP juga kompatibel dengan *web server* yang banyak digunakan sekarang seperti Apache, IIS (*Internet Information Service),* Caudium, Xitami, Omni, dan *web server* lainnya. PHP juga mampu berkomunikasi hampir dengan semua sistem basis data yang ada sekarang, seperti MySQL, PostgreSQL, Oracle, dan lain-lain. *Script* PHP dieksekusi di komputer *server* dimana *script* tersebut dijalankan, kemudian hasilnya dikirim ke *web browser client*. PHP membuat sebuah halaman *web* menjadi lebih dinamis, lebih interaktif dan halaman yang ditampilkan dibuat saat *client* melakukan *request* halaman tersebut sehingga informasi yang diterima oleh *client* adalah informasi yang baru. Pada sistem ini versi PHP yang digunakan adalah versi PHP 5.2.8.

2.6.2 MySQL

MySQL adalah sebuah aplikasi RDBMS (*Relational Data Base Management System*) yang sangat cepat dan kuat dalam menangani basis data. MySQL adalah sebuah *server* basis data yang dapat menangani banyak pengguna dan banyak tugas dalam waktu yang bersamaan. MySQL ini menggunakan bahasa SQL (*Structured Query Language*) yaitu sebuah bahasa *query* basis data standar dunia.

Menurut Al-Bahra Bin Ladjamudin (2005: 129) terdapat beberapa definisi basis data dari beberapa orang ahli basis data adalah sebagai berikut:

1. *Database* adalah sekumpulan data *store* (bisa dalam jumlah yang sangat besar) yang tersimpan dalam *magnetic disk, optical disk, magnetic drum* atau media penyimpanan sekunder lainnya.

2. *Database* adalah sekumpulan program-program aplikasi umum yang bersifat "*batch*" yang mengeksekusi dan memproses data secara umum (seperti pencarian, peremajaan, penambahan dan penghapusan terhadap data).
3. *Database* terdiri dari data yang akan digunakan atau diperuntukan terhadap banyak *user*, di mana masing-masing *user* (baik menggunakan teknik pemrosesan yang bersifat *batch* atau *on-line*) akan menggunakan data tersebut sesuai dengan tugas dan fungsinya, dan *user* lain dapat juga menggunakan data tersebut dalam waktu yang bersamaan.
4. *Database* adalah koneksi terpadu dari data-data yang saling berkaitan dari suatu *enterprise* (perusahaan, instansi pemerintah atau swasta).

Kelebihan MySQL dibandingkan jenis basis data lainnya di antaranya:

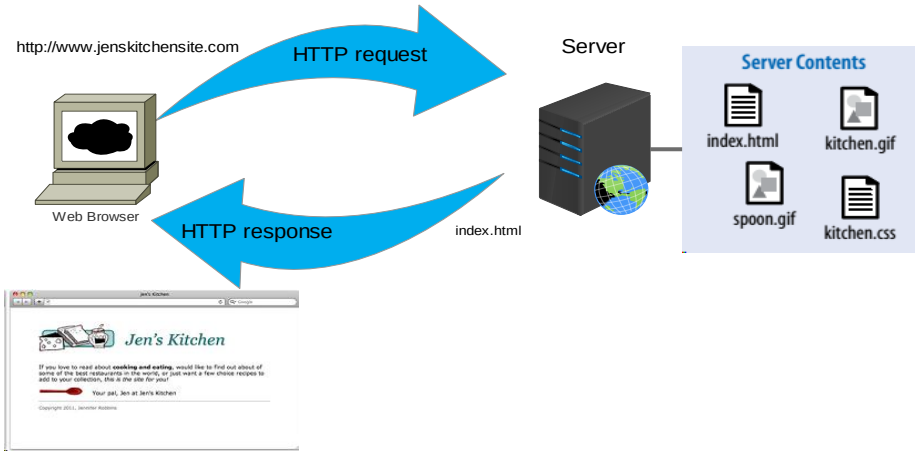
1. Bersifat terbuka dalam penggunaannya (*open source*).
2. Menggunakan bahasa pemrograman yang umum dan mirip dengan yang digunakan oleh bahasa pemrograman basis data yang lain.
3. Terdapat banyak literatur untuk mempelajari MySQL sehingga membuat MySQL mudah untuk dipelajari.
4. MySQL dapat melakukan pemrosesan basis data yang banyak dengan waktu yang sangat cepat.
5. Dapat menyimpan *Record* dalam jumlah yang sangat besar (lebih dari 50 juta *record*, 60 ribu tabel serta 5 milyar baris).
6. Memiliki sistem *user priviledge* yang mudah dan efisien.
7. Kompabilitas dengan berbagai sistem operasi dan *web server* yang ada.
8. Memiliki perintah yang memudahkan dalam pemanipulasian atau pengaksesan data.
9. Memiliki struktur tabel yang fleksibel

2.6.3 Pemrograman *web*

World Wide Web (WWW) yang lebih dikenal dengan web, merupakan salah satu layanan yang didapat oleh pemakai komputer yang terhubung ke Internet.Web pada awalnya adalah ruang informasi dalam Internet, dengan menggunakan teknologi *hypertext*, pemakai dituntun untuk menemukan informasi dengan mengikuti *link* yang disediakan dalam dokumen web yang ditampilkan dalam *web browser*. Internet identik dengan web, karena popularitasnya sebagai penyedia informasi dan tampilan antar muka (*interface*) yang dibutuhkan oleh pengguna Internet

dari masalah informasi sampai dengan komunikasi. Web memudahkan pengguna komputer untuk berinteraksi dengan pelaku Internet lainnya dan menelusuri informasi. Selain itu web telah diadopsi oleh perusahaan sebagai bagian dari strategi teknologi informasinya, karena beberapa alasan yaitu akses informasi mudah, *set-up server* lebih mudah, informasi mudah didistribusikan, dan bebas *platform*, yaitu informasi dapat disajikan oleh *web browser* pada sistem operasi mana saja karena adanya standar dokumen berbagai tipe data dapat disajikan

Server dan *web browser* berkomunikasi satu sama lain dengan *protocol* yang memang dibuat khusus untuk ini, yaitu HTTP *(Hypertext Transfer Protocol)* bertugas menangani permintaan-permintaan (*request*) dari *browser* untuk mengambil dokumen–dokumen web. Berikut skema kerja antara *server* dan *web browser* .



Gambar 2.14 Skema Kerja Web (Robbins, 2012)

Aplikasi web (*web application*) adalah aplikasi yang dapat diakses dengan menggunakan *web browser* lewat jaringan baik internet ataupun intranet (misal intranet perusahaan). Halaman-halaman web yang telah ditambahkan kode program (PHP, ASP, JSP, Perl, dsb) biasa dikenal dengan nama *web application* (aplikasi web). Saat ini terdapat berbagai macam aplikasi web, diantaranya adalah *webmail*, *online shopping*, *blog*, *search engine* (mesin pencarian), SFA (*Sales Force Automation*), ERP (*Enterprise Resource Planning*), *online auction* (lelang online), CRM (*Customer Relationship Management*), berbagai Sistem Informasi suatu organisasi dan sebagainya.

2.7 Analisa dan Perancangan Berorientasi Objek dengan *Unified Modeling Language*

1.7.1 Konsep dasar analisa dan perancangan berorientasi objek

Menurut Dennis (2009), analisis sistem mendeskripsikan apa yang harus dilakukan oleh sistem untuk memenuhi kebutuhan informasi pengguna. Analisis sistem akan menjawab pertanyaan siapa yang akan menggunakan sistem, apa yang akan dikerjakan oleh sistem, dan dimana serta kapan sistem tersebut akan digunakan. Sedangkan perancangan sistem menentukan bagaimana sistem akan memenuhi tujuan tersebut, dalam hal ini: perangkat keras, perangkat lunak, infrastruktur jaringan; antarmuka pengguna, formulir dan laporan; serta program-program khusus, database, dan file yang akan dibutuhkan.(Dennis et al. 2009)

Konsep *object oriented* atau berorientasi obyek memfokuskan pada penciptaan *class* yang merupakan *blueprint* dari suatu objek. Konsep ini membagi perangkat lunak menjadi beberapa objek yang saling berinteraksi antara satu dengan lainnya. Beberapa istilah yang berkaitan dengan konsep *object oriented* adalah:

1. *Class* dan *Object*. *Class* dapat diartikan deskripsi secara umum (*template*, *pattern*, atau *blueprint*) yang menggambarkan sekumpulan *object* yang serupa. *Object* dapat berupa *object* fisik seperti meja atau pelanggan maupun *object* konseptual seperti *text input area* atau file.
2. *Attribute, Method* dan *Message*. *Attribute* adalah sesuatu yang melekat pada *object* yang mendeskripsikan sifat *class* atau *object*. Sebuah *object* mengenkapsulasi data (direpresentasikan sebagai kumpulan *attribute*) dan algoritma yang memproses data tersebut. Algoritma ini disebut operasi, *method*, atau *service*. Setiap operasi yang dienkapsulasi oleh sebuah *object* memberikan representasi salah satu *behavior* dari *objek* tersebut. Suatu *object* berinteraksi dengan *object* lainnya melalui *message*. Sebuah *object* diminta untuk melakukan salah satu operasinya dengan mengirimkannya sebuah *message*. *Object* penerima merespon *message* tersebut dengan memilih operasi yang mengimplementasikan nama *message*, mengeksekusi operasi, dan kemudian mengembalikan kontrol kepada *object* yang memanggil.
3. Enkapsulasi. Sebuah *class* mengenkapsulasi data dan operasi yang memproses data tersebut. Data atau *attribute* yang menggambarkan kelas ditutup oleh operasi yang memanipulasi data tersebut.
4. *Inheritance*. *Inheritance* merupakan pewarisan sifat dari sebuah *class* ke *class* yang baru. *Subclass* Y merupakan pewaris dari *superclass* X, maka *subclass* Y mewarisi semua *atribut* dan operasi yang dimiliki oleh *superclass* X.

5. *Polimorfisme*. *Polimorfisme* mengijinkan sejumlah operasi yang berbeda untuk mempunyai nama yang sama. Hal ini membuat *object* saling terpisah dari *object* lainnya dan membuat setiap *object* lebih independen.

1.7.2 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah bahasa pemodelan yang digunakan untuk menganalisis, menentukan, dan desain sistem software. Metodologi berorientasi objek mulai muncul dari akhir 1980-an hingga tahun 1990-an, banyak metodologi muncul dan kemudian dimodifikasi dan kembangkan. Banyak dari beberapa metodologi kuat di daerah tertentu namun lemah di sisi lain. Hal ini memberi kebangkitan bagi para methodologists untuk bisa mengadopsi aspek yang berguna dari metodologi lain kedalam metodologi yang mereka miliki.

Pertengahan tahun 1990, Booch, Rumbaugh, and Jacobson mulai begabung di perusahaan software rational dan mulai mencurahkan ide dari metodologi yang mereka miliki untuk membuat bentuk awal dari UML. Lalu mereka mulai bekerja sama dengan para metodologiest dan perusahaan untuk mengusulkan bahasa pemodelan yang standard ke *Object Management Group* (OMG). OMG adalah konsorsium yang menciptakan dan mempertahankan standard untuk industri komputer. Pada bulan November 1997, OMG mengadopsi UML sebagai standar. Sejak saat itu OMG telah diasumsikan kepengurusan dan berkelanjutan pengembangan UML. (Booch et al. 2007)

UML terus-menerus direvisi, sampai saat ini UML memiliki 3 kelompok diagram yaitu:

1. *Structure Diagram*

   Didalamnya terdapat: *package, class, objects, composite structure, component, profile, and deployment diagrams*. Diagram ini difungsikan untuk menentukan apa yang harus di terapkan didalam sistem. Diagram ini membantu untuk menentukan bagian dari arsitektur sistem

2. *Behavior Diagram*

   Di dalamnya terdapat : *use case, activity, and state machine diagrams*. Diagram ini menekankan apa yang harus terjadi didalam system atau proses bisnis. Diagram ini berfungsi untuk menggambarkan fungsi dari system.

3. *Interaction Diagram*

Terdiri dari: *communication, sequence, timing, and interaction overview*. Ini adalah bagian dari behavior diagram yang menggambarkan aliran control antara komponen yang berbeda didalam system.

UML menyediakan beberapa notasi dan diagram standar yang dapat digunakan sebagai alat komunikasi bagi para pengembang sistem dalam proses analisis dan desain sistem. Tidak setiap diagram harus digunakan selama pengembangan sistem. Hanya diagram yang bisa mewakili informasi yang berguna untuk proyek yang direkomendasikan. (Wazlawick 2013)

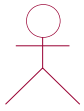
Berdasarkan perspektif dalam proses analisis dan perancangan berorientasi obyek dengan UML, terdapat beberapa diagram utama dalam UML yang dapat digunakan, yaitu:

a. Proses Analisis

1. *Use Case Diagram*

   Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. *Use Case* juga menggambarkan interaksi yang terjadi dalam sistem, interaksi itu antara sistem di dalam dengan sistem di luar dan *user* atau *actor*, yang memberi gambaran *user* atau *actor* yang berhubungan dengan sistem dan hal-hal yang berhubungan dengan user di dalam sistem.(Dennis et al. 2009)

Tabel 2.4 Elemen-elemen *Use Case Diagram*

Nama Elemen	Fungsi	Notasi
Actor	Menggambarkan orang atau sistem yang berhubungan dengan sistem dan dengan subyek di luarnya, diletakkan di luar pembatas subyek, dan dapat diasosiasikan dengan *actor* lain dengan menggunakan *specialization* atau *superclass association*.	 Actor/Role
Use Case	Mewakili sebuah bagian dari fungsionalitas sistem dan ditempatkan dalam *system boundary*.	 Use Case
Subject boundary	Menggambarkan lingkup subyek.	Subject
Assocation Relationship	Menggambarkan hubungan antara *actor* dengan *use case*.	* *

Include relationship	Menggambarkan hubungan ke dalam sistem. Arah panah dari *base use case* ke *included use case*.	<<include>>
Extend relationship	Menggambarkan hubungan dengan pilihan *optional*. Arah panah dari *extension use case* ke *base use case*.	<<extend>>
Generalization Relationship	Menggambarkan hubungan dalam sistem, antara satu *use case* dengan *use case* lain.	

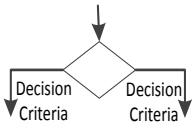
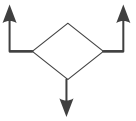
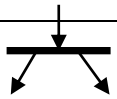
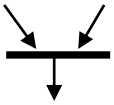
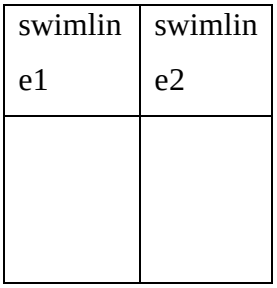
2. *Activity Diagram*

Activity Diagram merupakan model analisis yang digunakan atau menggambarkan sebuah proses aktivitas. Diagram ini dapat dipakai untuk berbagai model proses. Beberapa kegunaan dari *activity diagram* antara lain:

a. Memodelkan suatu proses atau operasi.
b. Untuk menggambarkan sebuah fungsi sistem.
c. Dalam sebuah operasi yang spesifik, diagram ini dipakai untuk menggambarkan logika dari sebuah proses atau operasi.

Tabel 2.5 Elemen-elemen *Activity Diagram*

Nama Elemen	**Fungsi**	**Notasi**
Action	Menggambarkan aksi atau aktivitas dari suatu proses.	Action
Activity	Menggambarkan koneksi dalam sebuah aliran proses.	Activity
Object Node	Mewakili *objek* yang terhubung dengan kumpulan *object flow*.	Class
Control Flow	Menunjukkan aliran sebuah objek dari satu aktivitas atau aksi ke aktivitas atau aksi lainnya.	
Initial Node	Menggambarkan proses mulai berjalan atau *start*.	
Final-	Untuk menghentikan semua proses kontrol	

Nama Elemen	Fungsi	Notasi
Activity Node	atau aliran *objek* pada sebuah aktivitas atau aksi.	
Final-Flow Node	Untuk menghentikan *control flow* atau *object flow* tertentu.	
Desicion Node	Ini digunakan untuk menggambarkan suatu kondisi untuk mengambil keputusan.	
Merge Node	Untuk menyatukan kembali *decision path* yang dibuat dengan menggunakan *decision node*.	
Fork Node	Untuk memisahkan atau membagi menjadi sepasang aksi atau aktivitas yang berjalan bersamaan.	
Join Node	Untuk menyatukan kembali kumpulan aktivitas yang berjalan secara paralel atau bersamaan menjadi satu aktivitas atau aksi.	
Swimlane	Untuk membagi sebuah *activity diagram* menjadi kolom guna menempatkan aktivitas atau aksi tertentu pada individu atau *objek* yang bertanggung jawab untuk melaksanakan aktivitas atau aksi tersebut.	

3. *Sequence Diagram*

Sequence diagram menggambarkan *objek* yang ada dalam *use case* dan *message* yang berjalan dalam suatu *use case*. Diagram ini juga menggambarkan *objek* dan relasinya termasuk kronologi (urutan) perubahan secara logis setelah menerima sebuah *message*.

Tabel 2.6 Elemen-elemen *Sequence Diagram* (Dennis et al, 2009)

Nama Elemen	**Fungsi**	**Notasi**
Actor	Merupakan orang atau sistem yang memiliki hubungan dengan sistem dan berada di luar ke sistem, hubungan dalam *sequence* ini diperlihatkan dengan mengirim atau menerima *message*.	 Actor/Role
Object	Simbol ini juga diletakkan di atas, dan hubungan dalam *sequence* juga diperlihatkan dengan mengirim atau menerima *message*.	Obiect :
Lifeline	Menandakan hidup dari *objek* dalam *sequence*.	
Execution Occurrence	Menandakan sebuah *objek* yang sedang mengirim atau menerima *message*.	
Message	Untuk menyampaikan informasi dari satu *objek* ke *objek* lain.	message() return value

b. Proses Perancangan

1. *Class Diagram*

   Class diagram menggambarkan sejumlah *class* dan hubungan antar *class* tersebut di dalam sistem. Selama perancangan, *class diagram* digunakan untuk meng-*capture* struktur *class* yang membangun arsitektur sistem. Dua elemen utama dari *class diagram* adalah *class* dan *relationship*. (Dennis et al. 2009)

2. *Deployment Diagram*

Deployment Diagram digunakan untuk mewakili hubungan antara komponen hardware yang digunakan dalam infrastruktur fisik sistem informasi. *Deployment Diagram* juga dapat digunakan untuk mewakili komponen perangkat lunak dan bagaimana komponen tersebut ditempatkan di atas arsitektur fisik atau infrastruktur sistem informasi. (Dennis et al. 2009).

Tujuan penggunaan berbagai jenis diagram UML tersebut adalah:

1. UML menunjukan semua spesifikasi analisis, perancangan dan implementasi yang penting dan dibuat pada saat pengembangan sistem. Diagram yang berbeda-beda tersebut dapat menyatakan tingkatan yang berbeda dalam proses rekayasa.
2. Model UML dapat dikoneksikan secara langsung pada bahasa pemograman visual. Maksudnya membangun model yang dapat di-*mapping* ke bahasa pemograman atau tabel pada database relational atau penyimpanan tetap pada database berorientasi object.
3. Dengan diagram diharapkan dapat membuat model sistem yang semakin mendekati realitas.
4. UML menunjukkan dokumentasi dari arsitektur sistem dan detail dari sistem yang dibangun.

1.7.3 *Software testing*

Software testing merupakan perangkat lunak diuji untuk mengungkap kesalahan yang dibuat secara tidak sengaja pada saat perangkat lunak itu dirancang atau dibangun. (Roger Pressman, 2012)

Pengujian perangkat lunak adalah sekumpulan langkah yang dimana anda dapat menempatkan sebuah teknik rancangan kasus terhadap pengujian tertentu dengan metode pengujian sebaiknya didefinisikan dalam proses perangkat lunak.

1.7.3.1 Teknik pengujian

Terdapat dua jenis pendekatan kasus uji yaitu *White-box* and *Black-box* pendekatan *white-box* adalah pengujian untuk memperlihatkan cara kerja dari sebuah produk secara rinci sesuai dengan spesifikasinya. Jalur logika perangkat lunak akan diuji coba dengan menyediakan kasus uji yang akan mengerjakan kumpulan kondisi dan pengulangan secara spesifik, sehingga melalui penggunaan metode ini akan dapat memperoleh kasus uji yang menjamin bahwa semua jalur independen pada suatu model telah digunakan minimal satu loop dalam batasan dan batas operasional perekayasa, serta pengginaan struktur data internal guna menjamin validasainya,

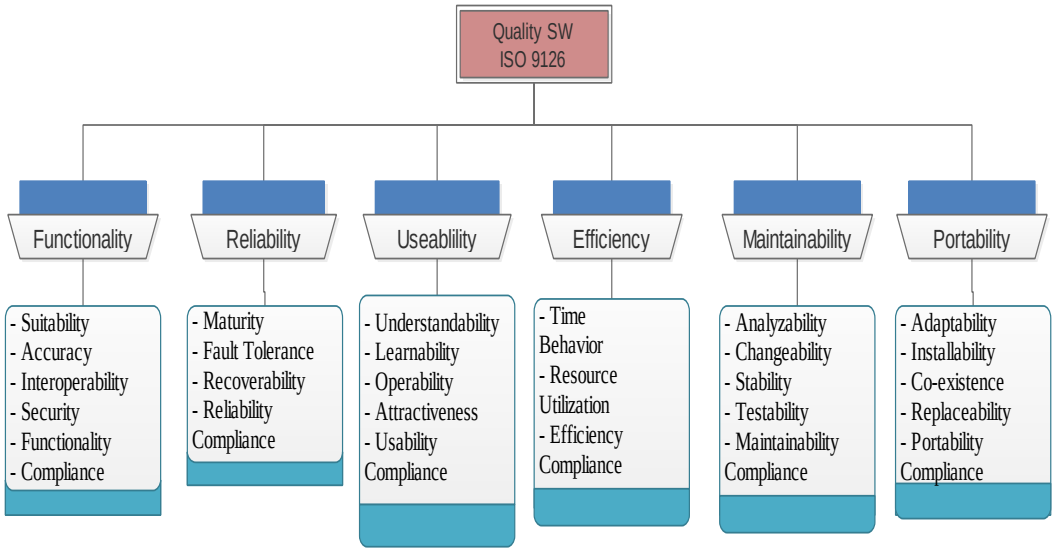
secara sekilas dapat diambil sebuah kesimpulan dengan pendekatan pengujian white-box mengarah untuk mendapatkan program yang benar secara 100%.

Pendekatan black-box merupakan pengujian untuk mengetahui apakah semua fungsi perangkat lunak telah berjalan semestinya sesuai dengan kebutuhan fungsional yang telah didefinisikan.

1.7.3.2 ISO 9126

Kualitas perangkat lunak dapat dinilai melalui ukuran-ukuran dan metode-metode tertentu, serta melalui pengujian-pengujian software. Salah satu tolak ukur kualitas perangkat lunak adalah ISO 9126, yang dibuat oleh *International Organization for Standardization* (ISO) dan *International Electrotechnical Commission* (IEC). ISO 9126 mendefinisikan kualitas produk perangkat lunak, model, karakteristik mutu, dan metrik terkait yang digunakan untuk mengevaluasi dan menetapkan kualitas sebuah produk software. Standar ISO 9126 telah dikembangkan dalam usaha untuk mengidentifikasi atribut-atribut kunci kualitas untuk perangkat lunak komputer. Faktor kualitas menurut ISO 9126 meliputi enam karakteristik kualitas sebagai berikut: (Al-Qutaish 2010)

1. *Functionality* (Fungsionalitas). Kemampuan perangkat lunak untuk menyediakan fungsi sesuai kebutuhan pengguna, ketika digunakan dalam kondisi tertentu.
2. *Reliability* (Kehandalan). Kemampuan perangkat lunak untuk mempertahankan tingkat kinerja tertentu, ketika digunakan dalam kondisi tertentu.
3. *Usability* (Kebergunaan). Kemampuan perangkat lunak untuk dipahami, dipelajari, digunakan, dan menarik bagi pengguna, ketika digunakan dalam kondisi tertentu.
4. *Efficiency* (Efisiensi). Kemampuan perangkat lunak untuk memberikan kinerja yang sesuai dan relative terhadap jumlah sumber daya yang digunakan pada saat keadaan tersebut.
5. *Maintainability* (Pemeliharaan). Kemampuan perangkat lunak untuk dimodifikasi. Modifikasi meliputi koreksi, perbaikan atau adaptasi terhadap perubahan lingkungan, persyaratan, dan spesifikasi fungsional
6. *Portability* (Portabilitas). Kemampuan perangkat lunak untuk ditransfer dari satu lingkungan ke lingkungan lain



Gambar 2.15 Karakteristik dan sub Karakteristik (Al-Qutaish 2010)

Masing-masing karakteristik kualitas perangkat lunak model ISO 9126 dibagi menjadi beberapa sub-karakteristik kualitas, yaitu:

Tabel 2.7 Karakteristik dan Sub Karakteristik ISO 9126 (Al-Qutaish 2010)

Karakteristik	**Sub-karakteristik**	**Deskripsi**
Functionality	*Suitability*	Kemampuan perangkat lunak untuk menyediakan serangkaian fungsi yang sesuai untuk tugas-tugas tertentu dan tujuan pengguna.
	Accuracy	Kemampuan perangkat lunak dalam memberikan hasil yang presisi dan benar sesuai dengan kebutuhan.
	Security	Kemampuan perangkat lunak untuk mencegah akses yang tidak diinginkan, menghadapi penyusup (*hacker*) maupun otorisasi dalam modifikasi data.
	Interoperability	Kemampuan perangkat lunak untuk berinteraksi dengan satu atau lebih sistem tertentu.
	Compliance	Kemampuan perangkat lunak dalam memenuhi standar dan kebutuhan sesuai peraturan yang berlaku.
Reliability	*Maturity*	Kemampuan perangkat lunak untuk menghindari kegagalan sebagai akibat dari kesalahan dalam perangkat lunak.
	Fault tolerance	Kemampuan perangkat lunak untuk mempertahankan kinerjanya jika terjadi kesalahan perangkat lunak
	Recoverability	Kemampuan perangkat lunak untuk membangun kembali tingkat kinerja dan memulihkan data yang rusak.
Usability	*Understandibility*	Kemampuan perangkat lunak dalam kemudahan untuk dipahami.

Karakteristik	**Sub-karakteristik**	**Deskripsi**
	Learnability	Kemampuan perangkat lunak dalam kemudahan untuk dipelajari.
	Operability	Kemampuan perangkat lunak dalam kemudahan untuk dioperasikan.
	Attractiveness	Kemampuan perangkat lunak dalam menarik pengguna.
Efficiency	*Time behavior*	Kemampuan perangkat lunak dalam memberikan respon dan waktu pengolahan yang sesuai saat melakukan fungsinya.
	Resource behavior	Kemampuan perangkat lunak dalam menggunakan sumber daya yang dimilikinya ketika melakukan fungsi yang ditentukan.
Maintainability	*Analyzability*	Kemampuan perangkat lunak dalam mendiagnosis kekurangan atau penyebab kegagalan.
	Changeability	Kemampuan perangkat lunak untuk dimodifikasi tertentu.
	Stability	Kemampuan perangkat lunak untuk meminimalkan efek tak terduga dari modifikasi perangkat lunak.
	Testability	Kemampuan perangkat lunak untuk dimodifikasi dan divalidasi perangkat lunak lain.
Portability	*Adaptability*	Kemampuan perangkat lunak untuk diadaptasikan pada lingkungan yang berbeda-beda.
	Instalability	Kemampuan perangkat lunak untuk diinstal dalam lingkungan yang berbeda-beda.
	Coexistence	Kemampuan perangkat lunak untuk berdampingan dengan perangkat lunak lainnya dalam satu lingkungan dengan berbagi sumber daya.
	Replaceability	Kemampuan perangkat lunak untuk digunakan sebagai sebagai pengganti perangkat lunak lainnya.

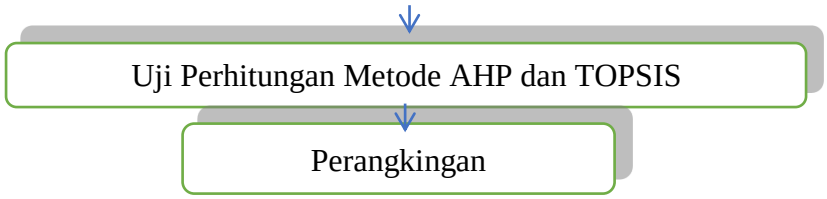
2.8 Tinjauan Studi

Beberapa jurnal menjelaskan tentang penerapan *Fuzzy* AHP dan *Fuzzy* TOPSIS dalam pengambilan sebuah keputusan diantaranya adalah:

1. Penelitian oleh Novianto Dwi Prasongko dan Rahmat Gernowo pada tahun 2015 dengan judul Metode Quality Function Deployment (QFD) dan Fuzzy TOPSIS Untuk Sistem Pendukung Keputusan Pemilihan Perusahaan Penyedia Jasa Internet. Penelitian ini membahas tentang Sistem Pendukung Keputusan Untuk memilih perusahaan *Internet Service Provider* (ISP) atau penyedia jasa internet terbaik berdasarkan perigkat dari pengolahan metode *fuzzy* TOPSIS, bobot kriteria diberikan terlebih dahulu berdasarkan survey QFD dan masih terbatas untuk jumlah kriteria dan alternatif.
2. Selanjutnya penelitian oleh Juliyanti, Mohammad Isa Irawan, dan Imam Mukhlaas pada tahun 2011 dengan judul "Pemilihan Guru Berprestasi Menggunakan Metode AHP dan TOPSIS". Penelitian ini membangun suatu model pengambilan keputusan multikriteria dengan menggunakan metode AHP untuk menentukan bobot dari kriteria yang telah ditentukan dan kemudian melakukan perankingan alternatif dengan menggunakan metode TOPSIS pada pemilihan guru berprestasi tingkat SLTA. Kelemahan tidak menggunakan bilangan fuzzy sehingga jarak alternatif tdk bisa dibuat lebih kecil.
3. Penelitian oleh Sriyanto, Wiwik Budiawan, dan Farhania Aisyah Setiowati dengan judul "Studi Penerapan Metode Fuzzy Ahp Dan Topsis Untuk Evaluasi Preferensi Moda Transportasi Umum Di Kota Semarang". Penelitian ini membahas tentag pemiliha moda transportasi umum terbaik untuk mengatasi masalah kemacetan di Kota Semarang. Fuzzy AHP digunakan untuk bobot prioritas alternatif yang mengindikasikan kepentingan dari kriteria dan sub-kriteria pada bobot pemilihan alternatif. Pengolahan data menggunakan TOPSIS.
4. Selanjutnya ada penelitian Büyüközkan dan Gizem (2012) dengan judul "*A combined fuzzy AHP and fuzzy TOPSIS based strategic analysis of electronic service quality in healthcare industry*" tujuan dari penelitian ini adalah, untuk menggunakan teknik multi kriteria hibrida yang menggabungkan *fuzzy* AHP dan *fuzzy* TOPSIS untuk mengevaluasi satu set alternatif situs *website* rumah sakit untuk mencapai alternatif terbaik.

5. Penelitian Kutlu dan Mehmet (2012) dengan judul "*Fuzzy failure modes and effects analysis by using fuzzy TOPSIS-based fuzzy AHP*" tujuan penelitian ini yaitu menganalisis *Failuremode and effects analysis*(FMEA). Untuk setiap mode kegagalan dapat diprediksi tiga faktor risiko yaitu tingkat keparahan (*Severity*), kejadian (*Occurence*), dan kemampuan mendeteksi (*Detectability*) dan dianalisis menggunakan nomor prioritas risiko (*RiskPriorityNumber*) dengan mengalikan faktor-faktor tersebut. Pendekatan *fuzzy* menggunakan variabel linguistik untuk menentukan S, O, dan D, dipertimbangkan untuk menganalisis FMEA dengan *fuzzy* TOPSIS dan *fuzzy* AHP.
6. Penelitian Taylan *et al* (2014) dengan judul "*Construction projects selection and risk assessment by fuzzy AHP and fuzzy TOPSIS methodologies*" Penelitian ini mengidentifikasi kriteria risiko proyek konstruksi pada King Abdul Aziz University (KAU), Metode *Relative Importance Index* (RII).

2.9 Kerangka Pemikiran



Gambar 2.16 Kerangka Pemikiran

2.10 Hipotesis Penelitian

Dugaan sementara dari penelitian ini adalah :

1. Metode *Analitical Hierarchy Process*(AHP) dapat digunakan untuk mencari nilai bobot dari setiap kriteria yang dibandingkan secara berpasangan dengan nilai bobot yang lebih akurat yaitu 80,3%, dan
2. Metode *Technique For Order Preference by Similarity to Ideal Solution* (TOPSIS) digunakan untuk proses perangkingan kandidat dari setiap kriteria alternatif dengan nilai akurasi yang tinggi yaitu 83,2% jika menggunakan kriteria penilaian yang lebih banyak dan nilai bobot yang diperoleh dari penghitungan *fuzzy* AHP.