BAB II

TINJAUAN PUSTAKA

II.1. Logika *Fuzzy*

Logika *fuzzy* merupakan salah satu komponen pembentuk *softcomputing*. Logika *fuzzy* pertama kali diperkenalkan oleh Prof. Lotfi A. Zadeh pada tahun 1965. Dasar logika *fuzzy* adalah teori himpunan *fuzzy*. Pada teori himpunan *fuzzy*, peranan derajad keanggotaan sebagai penentu keberadaan elemen dalam suatu himpunan sangatlah penting. Nilai keanggotaan atau derajad keanggotaan atau *membership function* menjadi cirri utama dari penalaran dengan logika *fuzzy* tersebut (Sri Kusumadewi dan Hari Purnomo; 2004: 1)

II.1.1. Himpunan *Fuzzy* (*Fuzzy Set*)

Pada himpunan tegas *(crisp)*, nilai keanggotaan suatu item x dalam suatu himpunan A, yang sering ditulis dengan $\mu_A(x)$, memiliki dua kemungkinan, yaitu:

1. Satu (1), yang berarti bahwa suatu item menjadi anggota dalam himpunan, atau
2. Nol (0), yang berarti bahwa suatu item tidak menjadi anggota dalam suatu himpunan.

Contoh II.1:

jika diketahui:

S= {1, 2, 3, 4, 5, 6} adalah semesta pembicaraan.

A= {1, 2, 3}

B= {3, 4, 5}

bisa dikatakan bahwa:

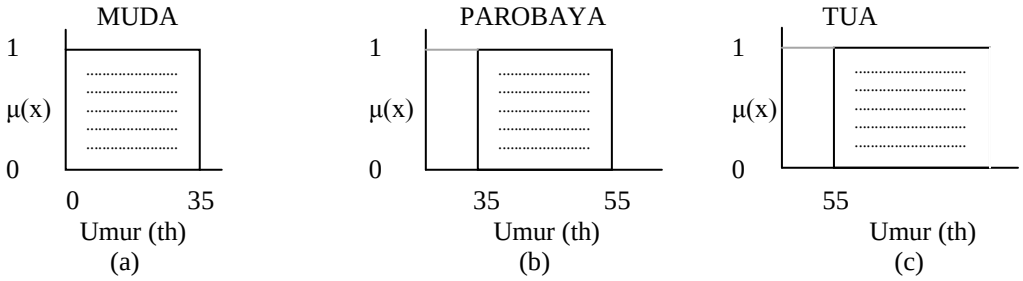
1. Nilai keanggotaan 2 pada himpunan A, $\mu_A(2) = 1$, karena $2 \in A$.
2. Nilai keanggotaan 3 pada himpunan A, $\mu_A(3) = 1$, karena $3 \in A$.
3. Nilai keanggotaan 4 pada himpunan A, $\mu_A(4) = 1$, karena $4 \notin A$.
4. Nilai keanggotaan 2 pada himpunan B, $\mu_B(2) = 1$, karena $2 \notin B$.
5. Nilai keanggotaan 3 pada himpunan B, $\mu_B(3) = 1$, karena $3 \in B$.

Contoh II.2:

Misalkan variabel umur dibagi menjadi 3 kategori, yaitu:

MUDA	umur < 35 tahun
PAROBAYA	35 ≤ umur ≤ 55 tahun
TUA	umur > 55 tahun

Nilai keanggotaan secara grafis, himpunan MUDA, PAROBAYA, dan tua ini dapat dilihat pada Gambar II.1.



Gambar II.1. Himpunan: MUDA, PAROBAYA, dan TUA

(Sumber: Sri Kusumadewi dan Hari Purnomo; 2004: 4)

Pada gambar II.1, dapat dijelaskan bahwa:

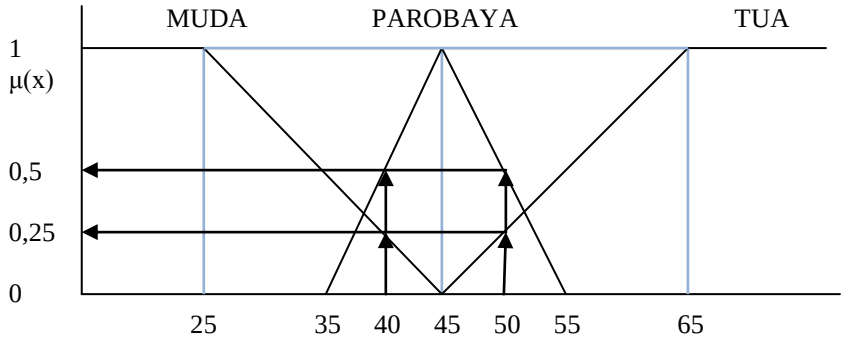
1. Apabila seseorang berusia 34 tahun, maka ia dikatakan MUDA ($\mu_{MUDA}(35)=1$);
2. Apabila seseorang berusia 35 tahun, maka ia dikatakan TIDAK MUDA ($\mu_{MUDA}(35)=0$);

3. Apabila seseorang berusia 35 tahun kurang 1 hari, maka ia dikatakan TIDAK MUDA (μ_{MUDA}(35 th – 1 hr)=0);
4. Apabila seseorang berusia 35 tahun, maka ia dikatakan PROBAYA ($\mu_{PAROBAYA}$(35)=1);
5. Apabila seseorang berusia 34 tahun, maka ia dikatakan TIDAK PAROBAYA ($\mu_{PAROBAYA}$(34)=0);
6. Apabila seseorang berusia 55 tahun, maka ia dikatakan PAROBAYA ($\mu_{PAROBAYA}$(55)=1);
7. Apabila seseorang berusia 35 tahun kurang 1 hari, maka ia dikatakan TIDAK PAROBAYA ($\mu_{PAROBAYA}$(35 th – 1 hr)=0);

Dari sini bisa dikatakan bahwa pemakaian himpunan crisp untuk menyatakan umur sanat tidak adil, ada nya perubahan kecil saja pada suatu nilai mengakibatkan perbedaan kategori yang cukup segnifikan.

Himpunan *fuzzy* digunakan untuk mengantisifasi hal tersebut. Seseorang dapat masuk dalam 2 himpunan yang berbeda, MUDA dan PAROBAYA, PAROBAYA dan TUA, dan sebagainya. Seberapa besar eksistensinya dalam himpunan tersebut dapat dilihat pada nilai keanggotaannya.

Gambar II.2 menunjukkan himpunan fuzzy untuk variabel umur.



Gambar II.2. Himpunan Fuzzy Untuk Variabel Umur

(Sumber: Sri Kusumadewi dan Hari Purnomo; 2004: 5)

Pada gambar II.2, dapat dilihat bahwa:

1. Seseorang yang berumur 40 tahun, termasuk dalam himpunan MUDA dengan $\mu_{MUDA}(40)=0,25$; namun dia juga termasuk dalam himpunan PAROBAYA dengan $\mu_{PAROBAYA}(40)=0,5$.
2. Seseorang yang berumur 50 tahun, termasuk dalam himpunan MUDA dengan $\mu_{TUA}(50)=0,25$; namun dia juga termasuk dalam himpunan PAROBAYA dengan $\mu_{PAROBAYA}(50)=0,5$.

Kalau pada himpunan crisp, nilai keanggotaan hanya ada 2 kemungkinan, yaitu 0 dan 1, pada himpunan *fuzzy* nilai keanggotaan terletak pada rentang 0 sampai 1. Apabila x memiliki nilai keanggotaan *fuzzy* $\mu_A(x)=0$ berarti x tidak menjadi anggota himpunan A, demikian pula apabila x memiliki nilai keanggotaan *fuzzy* $\mu_A(x)=1$ berarti x menjadi anggota penuh pada himpunan A.

Terkadang kemiripan antara keanggotaan *fuzzy* dengan probabilitas menimbulkan kerancuan. Keduanya memiliki nilai pada interval [0,1], namun interpretasi nilainya sangat berbeda antara kedua kasus tersebut. Keanggotaan *fuzzy* memberikan suatu ukuran terhadap pendapat atau keputusan, sedangkan

probabilitas mengidikasikan proporsi terhadap keseringan suatu hasil bernilai benar dalam jangka panjang. Misalnya, jika nilai keanggotaan suatu himpunan *fuzzy* MUDA adalah 0,9; maka tidak perlu dipermasalahkan berapa seringnya nilai itu diulang secara individual untuk mengharapkan suatu hasil yang hampir pasti muda. Dilain pihak, nilai probabilitas 0,9 muda berarti 10% dari himpunan tersebut diharapkan tidak muda.

Himpunan *fuzzy* memiliki 2 atribut, yaitu:

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

Ada beberapa hal yang perlu diketahui dalam memahami sistem *fuzzy*, yaitu:

1. Variabel *fuzzy*

   Variabel *fuzzy* merupakan variabel yang hendak dibahas dalam suatu sistem *fuzzy*. Contoh: umur, temperatur, permintaan, dan sebagainya.

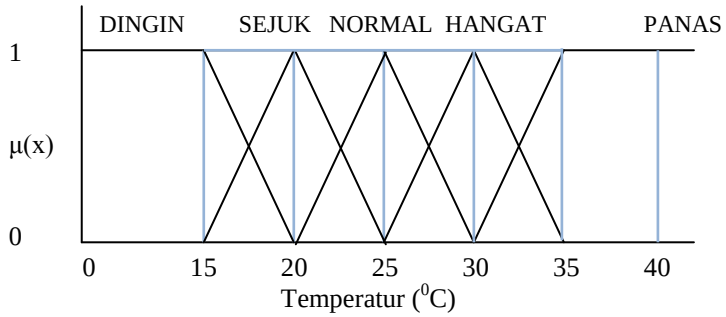
2. Himpunan *fuzzy*

   Himpunan *fuzzy* merupakan suatu grup yang mewakili suatu kondisi atau keadaan tertentu dalam suatu variabel *fuzzy*.

   Contoh:

   1. Variabel umur, terbagi menjadi 3 himpunan *fuzzy*, yaitu: MUDA, PAROBAYA, dan TUA.(Gambar II.2)

2. Variabel temperatur, terbagi menjadi 5 himpunan *fuzzy*, yaitu: DINGIN, SEJUK, NORMAL, HANGAT, dan PANAS.(Gambar II.3)



Gambar II.3. Himpunan *Fuzzy* pada Variabel Temperatur

(Sumber: Sri Kusumadewi dan Hari Purnomo; 2004: 7)

3. Semesta Pembicaraan

Semesta pembicaraan adalah keseluruhan nilai yang diperbolehkan untuk dioperasikan dalam suatu variabel *fuzzy*. Semesta pembicaraan merupakan himpunan bilangan real yang senantiasa naik (beratambah) secara monotan dari kiri ke kanan. Nilai semesta pembicaraan dapat berupa bilangan positif maupun nefatif. Adakalanya nilai semesta pembicaraan ini tidak dibatasi batas atasnya.

Contoh:

1. Semesta pembicaraan untuk variabel umur: [0 +∞)
2. Semesta pembicaraan untuk variabel temperatur: [0 40]

4. Domain

Domain himpunan *fuzzy* adalah keseluruhan nilai yang diizinkan dalam semesta pembicaraan dan boleh dioperasikan dalam suatu himpunan *fuzzy*. Seperti halnya semesta pembicaraan, domain merupakan himpunan bilangan real yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai domain dapat berupa nilai positif maupun negatif.

Contoh domain himpunan *fuzzy*:

1. MUDA = [0 45]
2. PAROBAYA = [35 55]
3. TUA = [45 +∞]
4. DINGIN = [0 20]
5. SEJUK = [15 25]
6. NORMAL = [20 30]
7. HANGAT = [25 35]
8. PANAS = [30 40]

II.2. Jenis- Jenis Fungsi Keanggotaan

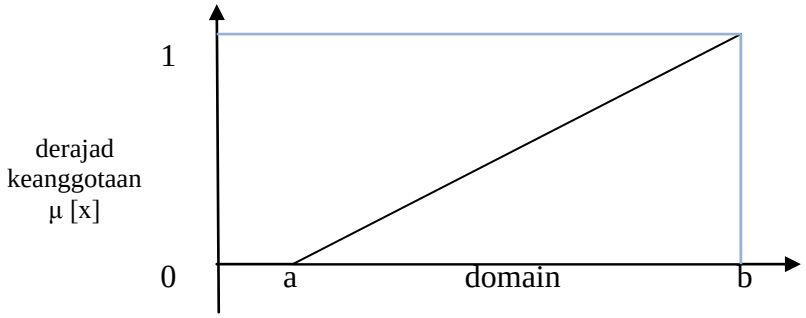
Fungsi keanggotaan (*member function*) adalah suatu kurva yang menunjukkan pemetaan titik- titik *input* data kedalam nilai keanggotaannya (sering juga disebut derajad keanggotaan) yang memiliki interval antara 0 sampai 1. Salah satu cara yang dapat digunakan untuk mendapatkan nilai keanggotaan adalah dengan melalui pendekatan fungsi.

II.2.1. Fungsi Representasi Linier

Pada representasi linier, pemetaan input ke derajat keanggotaannya digambarkan sebagai suatu garis lurus. Bentuk ini paling sederhana dan menjadi pilihan yang baik untuk mendekati suatu konsep yang kurang jelas. Keadaan linier himpunan *fuzzy* terdiri dari dua keadaan linier naik dan linier turun. Pada linier naik, kenaikan himpunan dimulai pada nilai domain yang memiliki derajat

keanggotaan nol [0] bergerak ke kanan menuju nilai domain yang memiliki derajat keanggotaan lebih tinggi dengan fungsi keanggotaan :

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

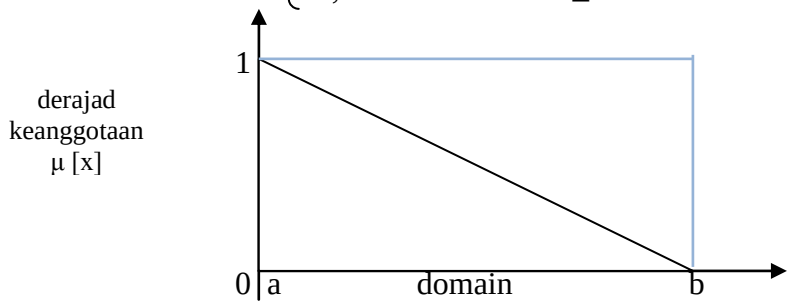


Gambar II.4. Fungsi Representasi Linier Naik

(Sumber : Sri Kusumadewi dan Hari Purnomo; 2004: 9)

Sedangkan pada linier turun, garis lurus dimulai dari nilai domain dengan derajat keanggotaan tertinggi pada sisi kiri, kemudian bergerak menurun ke nilai domain yang memiliki derajat keanggotaan lebih rendah dengan fungsi keanggotaan.

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



Gambar II.5. Fungsi Representasi Linier Turun

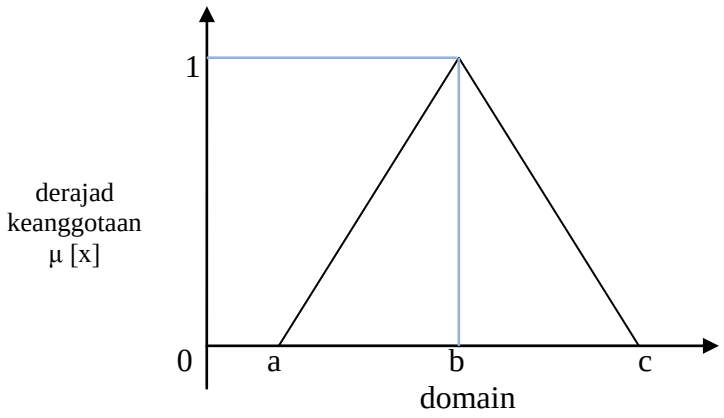
(Sumber : Sri Kusumadewi dan Hari Purnomo; 2004: 10)

II.2.2. Fungsi Keanggotaan Segitiga

Fungsi keanggotaan segitiga ditandai oleh adanya 3 (tiga) parameter {a,b,c} yang akan menentukan koordinat x dari tiga sudut. Kurva ini pada dasarnya merupakan gabungan antara dua garis (linier). Adapun persamaan untuk bentuk segitiga ini adalah :

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

Gambar gafik fungsi keanggotaan segitiga adalah:



Gambar II.6. Grafik Fungsi Keanggotaan Segitiga

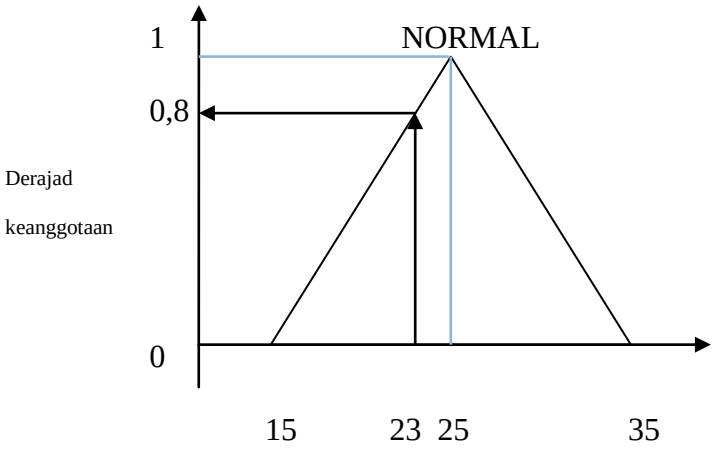
(Sumber : Sri Kusumadewi dan Hari Purnomo; 2004: 11)

Contoh II.3.

Fungsi keanggotaan untuk himpunan NORMAL pada variable temperature ruangan seperti terlihat pada gambar II.7.

μNORMAL(23) = (23-15)/(25-15)

= 8/10= 0.8



Gambar II.7. Himpunan *Fuzzy*: NORMAL (kurva segitiga)

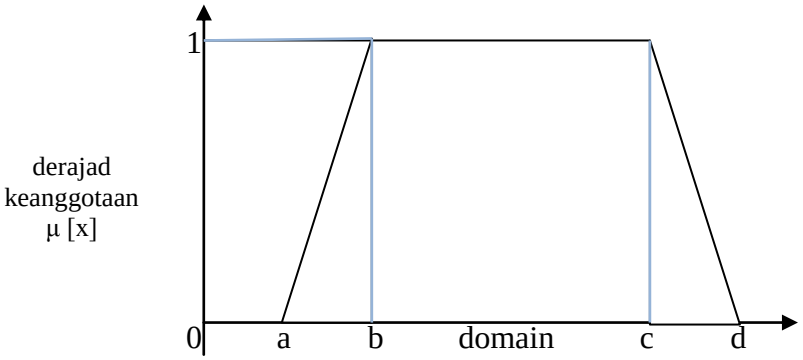
(Sumber : Sri Kusumadewi dan Hari Purnomo; 2004: 12)

II.2.3. Fungsi Keanggotaan Trapesium

Kurva trapesium pada dasarnya seperti bentuk segitiga, hanya saja ada beberapa titik yang memiliki nilai keanggotaan 1. Adapun persamaan untuk kurva trapesium ini adalah :

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

Adapun gambar grafik fungsi keanggotaannya adalah :



Gambar II.8. Grafik Fungsi Keanggotaan Trapesium

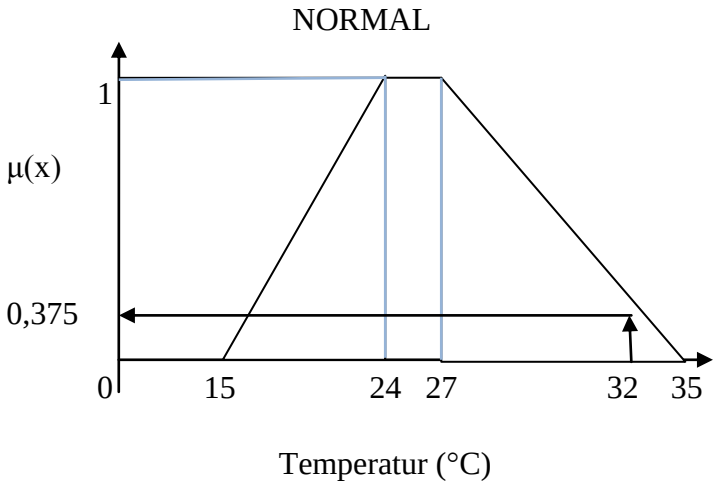
(Sumber : Sri Kusumadewi dan Hari Purnomo; 2004: 13)

Contoh II.4.

Fungsi keanggotaan untuk himpunan NORMAL pada variable temperature ruangan seperti terlihat pada gambar II.9.

μNORMAL (23) = (35-32)/(35-27)

= 3/8= 0,375

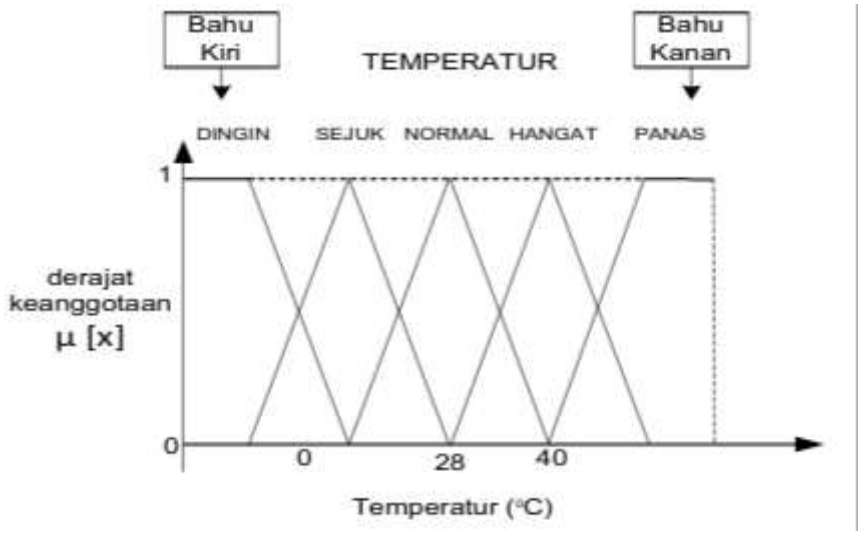


Gambar II.9. Himpunan *Fuzzy* NORMAL(Kurva Trapesium)

(Sumber : Sri Kusumadewi dan Hari Purnomo; 2004: 13)

II.2.4. Representasi Kurva Bahu

Representasi fungsi keanggotaan *fuzzy* dengan menggunakan kurva bahu pada dasarnya adalah gabungan dari kurva segitiga dan kurva trapesium. Daerah yang terletak di tengah-tengah suatu variabel yang direpresentasikan dalam bentuk segitiga, pada sisi kanan dan kirinya akan naik dan turun. Tetapi terkadang pada salah sisi dari variabel *fuzzy* yang ditinjau ini terdapat nilai yang konstan, yaitu pada himpunan ekstrim kiri dan ekstrim kanan. Hal ini dapat dilihat pada gambar II.8.



Gambar II.10. Representasi Kurva Bahu

(Sumber : Sri Kusumadewi dan Hari Purnomo; 2004: 14)

II.3. Operator Dasar Zadeh Untuk Operasi Himpunan *Fuzzy*

Seperti halnya himpunan konvensional, ada beberapa operasi yang didefinisikansecara khusus untuk mengkombinasi dan memodifikasi himpunan *fuzzy*. Nilai keanggotaan sebagai hasil dari operasi 2 himpunan sering dikenal

dengan nama *fire strength* atau α–predikat. Ada 3 operator dasar yang diciptakan oleh Zadeh, yaitu: (Sri Kusumadewi dan Hari Purnomo; 2004: 23)

II.3.1. Operator AND

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

$\mu_{A\cap B} = \min(\mu_A(x), \mu_B(y))$

Contoh II.5:

Misalkan nilai keanggotaan 27 tahun pada himpunan MUDA adalah 0,6 (μ_{MUDA} (27) = 0,6); dan nilai keanggotaan Rp 2.000.000,- pada himpunan penghasilan TINGGI adalah 0,8 ($\mu_{GAJITINGGI}$ ($2x10^6$) = 0,8); maka α–predikat untuk usia MUDA dan berpenghasilan TINGGI adalah:

$\mu_{MUDA\cap GAJITINGGI}$ = min (μMUDA (27),$\mu_{GAJITINGGI}$ ($2x10^6$))

= min (0,6; 0,8)

= 0,6

II.3.2. Operator OR

Operator ini berhubungan dengan operasi union pada himpunan.α-predikat sebagai hasil operasi dengan operator OR diperoleh dengan mengambil nilai keanggotaan terbesar antar elemen pada himpunan- himpunan yang bersanggutan.

$\mu_{A\cap B}$ = max (μ_A (x), μ_B (y))

Contoh II.6:

Pada contoh II.5, dapat dihitung nilai α-predikat untuk usia MUDA atau berpenghasilan TINGGI adalah:

$\mu_{MUDA \cap GAJITINGGI} = \max (\mu_{MUDA}(27), \mu_{GAJITINGGI}(2x10^6))$

$= \max (0,6; 0,8)$

$= 0,8$

II.3.3. Operator NOT

Operator ini berhubungan dengan operasi komplemen pada himpunan. α-predikat sebagai hasil operasi dengan operator NOT diperoleh dengan menggunakan nilai keanggotaan elemen pada himpunan yang bersangkutan dari 1.

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

Contoh II.7:

Pada contoh II.5, dapat dihitung nilai α-predikat untuk usia TIDAK MUDA adalah:

$\mu_{MUDA'}(27) = 1 - \mu_{MUDA}(27)$

$= 1 - 0,6$

$= 0,4$

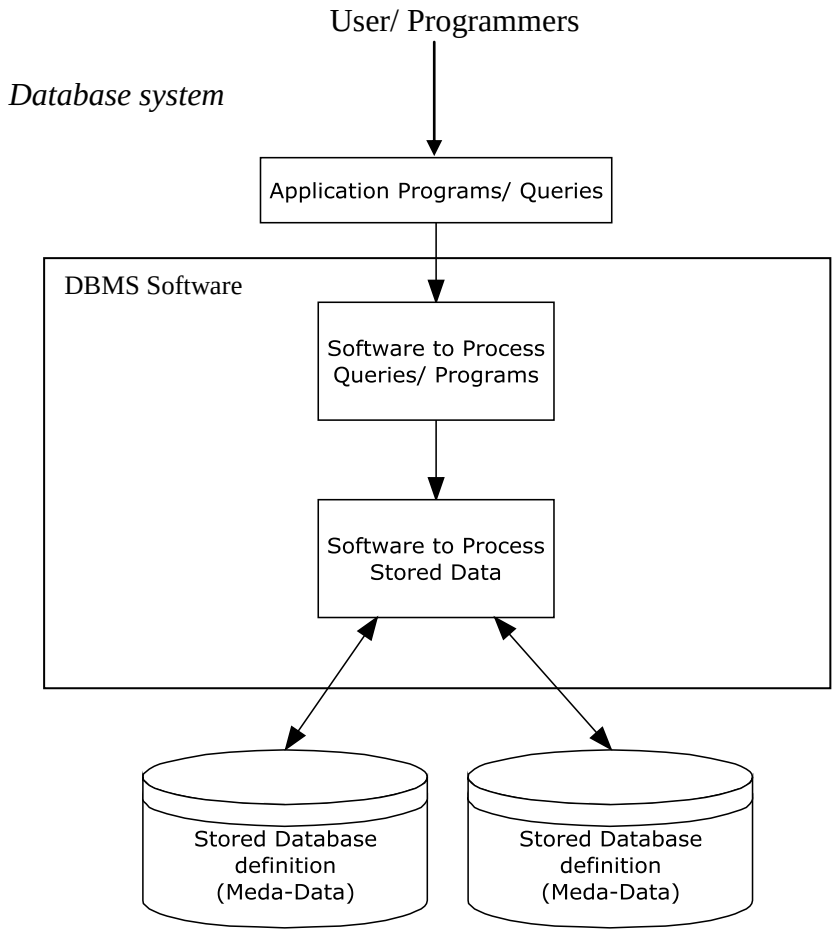
II.4. Database

Databse dan teknologi *database* memiliki pengaruh yang besar terhadap perkembangan penggunaan komputer. Dapat dikatakan bahwa *database* memainkan peranan yang penting pada hamper disegala bidang yang menggunakan computer, seperti bisnis, *electronic commerce (e-commerce), engineering*, kesehatan, hokum, edukasi dan masih banyak lagi.

Database adalah sekumpulan data yang saling berhubungan. Data adalah fakta yang dapat direkam dan memiliki arti secara implisit. Sebagai contoh, nama, nomor telepon dan alamat dari orang yang anda kenal. Anda dapat merekam data tersebut pada buku alamat atau anda simpan pada *harddisk* dengan menggunakan computer dan *software* aplikasiseperti Microsoft excel. Kumpulan data yang berhubungan dan memiliki arti secara implicit diatas disebut *database*.

Database Management System (DBMS) adalah sekumpulan *program* yang memungkinkan pengguna untuk membuat dan memelihara suatu *database*. Dengan kata lain, DBMS merupakan *general-purpose software system* yang memfasilitasi proses- proses seperti pengindefinisian, pembuatan, manipulasi aplikasi. Beberapa fungsi penting lainnya yang disediakan DBMS adalah keamanan dan pemeliharaan.

Database system tidak hanya berupa *database* itu sendiri tetapi juga terdiri dari definisi atau deskripsi struktur dan batasan- batasan (*constraints*) dari *database*. Definisi disimpan pada catalog DBMS, yang berisi informasi seperti struktur setiap file, tipe dan format penyimpanan dari setiap data dan berbagai batasan terhadap data tersebut. Informasi yang disimpan pada catalog DBMS disebut juga *meta-data*, yang menjelaskan struktur utama *database*. Gambar mengenai *database system* dapat dilihat seperti pada gambar II.11



Gambar II.11 Simplifikasi Lingkungan Database system

(Sumber: Setioyo Cahyono; 2006: 11)

II.4.1 Database *Relational*

Model relasional merupakan model yang paling sederhana sehingga mudah digunakan dan dipahami oleh pengguna, serta merupakan yang paling populer sat ini. Model inimenggunakan sekumpulan tabel berdimensi dua (yang disebut relasi atau tabel), dengan masing- masing relasi tersusun atas tupel atau baris dan atribut. Relasi dirancang sedemikian rupa sehingga dapat menhilangkan kemubaziran data dan menggunakan kunci tamu untuk berhubungan dengan relasi yang lain. Contoh

produk DBMS yang menggunakan model jaringan adalah CA-IDMS/ DB, dari *Computer Associates International Inc*. (sebelumnya dikenal sebagai IDMS – *Integrated Database Management System* – yang dikembangkan oleh *Cullient Software Inc*).

Tabel II.1 memperlihatkan istilah relasi, baris, dan atribut dan padanya dengan istilah- istilah lain yang populer dikalangan pemrograman dan sejumlah pengguna (terutama yang bekerja dengan SQL).

Tabel II.1 Padanan istilah relasi, tupel dan atribut

Model Relasional	Pemrogram	Pengguna
Relasi	Berkas	Tabel
Tupel (baris)	Rekaman	Baris
Atribut	Medan	Kolom

(Sumber: Abdul Kadir; 2003: 26)

II.5. *Fuzzy* Database

Basis data (*database*) merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan diperangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya. Sistem basis data (*database system*) adalah suatu sistem informasi yang mengintegrasikan kumpulan data yang saling berhubungan satu dengan yang lainnya dan membuatnya tersedia untuk beberapa aplikasi dalam suatu organisasi.

Basisdata yang umumnya kita gunakan, memiliki data yang lengkap dalam setiap tabelnya. Demikian pula, apabila hendak dibuat suatu *query*, maka *query* itupun harus menggunakan data yang ada pada tabel dan kata- kata kunci yang berlaku di SQL. Apabila kita memiliki data yang kurang lengkap, mengandung

ketidakpastian dan ambigu, maka pengguna basisdata biasa menjadi sulit untuk dilakukan. Dari sinilah, kita dapat memanfaatkan logika *fuzzy* untuk mengantisipasi pemanipulasian dalam basis data yang mengandung ketidakpastian, baik dari sisi data maupun *query*-nya. (Sumber: Sri Kusumadewi dan Hari Purnomo; 2004: 177)

II.5.1. *Fuzzy* Database Model Tahani

Sebagian besar basis data standar diklasifikasikan berdasarkan bagaimana data tersebut dipandang oleh *user*. Misalkan kita memiliki data karyawan yang tersimpan pada tabel DT_KARYAWAN dengan field NIP, nama, tgl lahir, th masuk, dan gaji per bulan seperti pada tabel II.2.

Tabel II.2 Data Mentah Karyawan

NIP	Nama	Tgl. Lahir	Thn. Masuk	Gaji/bln (Rp)
01	Lia	03-06-1972	1996	750.000
02	Iwan	23-09-1954	1985	1.500.000
03	Sari	12-12-1966	1988	1.225.000
04	Andi	06-03-1965	1998	1.040.000
05	Budi	04-12-1960	1990	950.000
06	Amir	18-11-1963	1989	1.600.000
07	Rian	28-05-1965	1997	1.250.000
08	Kiki	09-07-1971	2001	550.000
09	Alda	14-08-1967	1999	735.000
10	Yoga	17-09-1977	2000	860.000

(Sumber : Sri Kusumadewi dan Hari Purnomo; 2004: 178)

Kemudian dari tabel DT_KARYAWAN, kita olah menjadi suatu tabel temporer untuk menghitung umur karyawan dan masa kerjanya. Tabel tersebut kita beri nama dengan tabel KARYAWAN (Tabel II.3)

Tabel II.3 Data Karyawan Setelah Diolah

NIP	Nama	Umur (th)	Masa Kerja (th)*	Gaji/bln (Rp)
01	Lia	30	6	750.000

02	Iwan	48	17	1.500.000
03	Sari	36	14	1.225.000
04	Andi	37	4	1.040.000
05	Budi	42	12	950.000
06	Amir	39	13	1.600.000
07	Rian	37	5	1.250.000
08	Kiki	32	1	550.000
09	Alda	35	3	735.000
10	Yoga	25	2	860.000

(Sumber : Sri Kusumadewi dan Hari Purnomo; 2004: 178 dan 179)

Dengan menggunakan basisdata standar, kita dapat mencari data- data karyawan dengan klasifikasi tertentu dengan menggunakan *query*. Misal kita ingin mendapatkan informasi nama- nama karyawan yang usianya kurang dari 35 tahun, maka kita bisa ciptakan suatu *query*:

```
SELECT NAMA
FROM KARYAWAN
WHERE (Umur < 35)
```

Sehingga muncul nama- nama Lia, Kiki, dan Yoga. Apabila kita ingin mendapatkan informasi tentang nama- nama karyawan yang gajinya lebih dari 1 juta rupiah, maka kita bisa ciptakan suatu *query*:

```
SELECT NAMA
FROM KARYAWAN
WHERE (Gaji > 1000000)
```

Sehingga muncul nama- nama Iwan, Sari, Andi, Amir dan Rian. Apabila kita ingin mendapatkan informasi tentang nama- nama karyawan yang masa kerjanya kurang dari atau sama dengan 5 tahun tetapi gajinya sudah lebih 1 juta rupiah, maka kita bisa ciptakan *query*:

SELECT NAMA

FROM KARYAWAN

WHERE (MasaKerja ≤ 5) and (Gaji > 1000000)

Sehingga muncul nama- nama Andi dan Rian.

Pada kenyataannya, seseorang kadang membutuhkan informasi dari data-data yang bersifat *ambiguous*. Apabila hal ini terjadi, maka kita bisa mengatasinya dengan menggunakan basisdata *fuzzy*. Selama ini, sudah ada beberapa penelitian tentang basisdata *fuzzy*. Salah satunya adalah model Tahani. Basisdata fuzzy model Tahanimasih tetap menggunakan relasi standar, hanya saja model ini menggunakan teori himpunan *fuzzy* untuk mendapatkan informasi pada *query*-nya.

II.6. Sistem Pendukung Keputusan

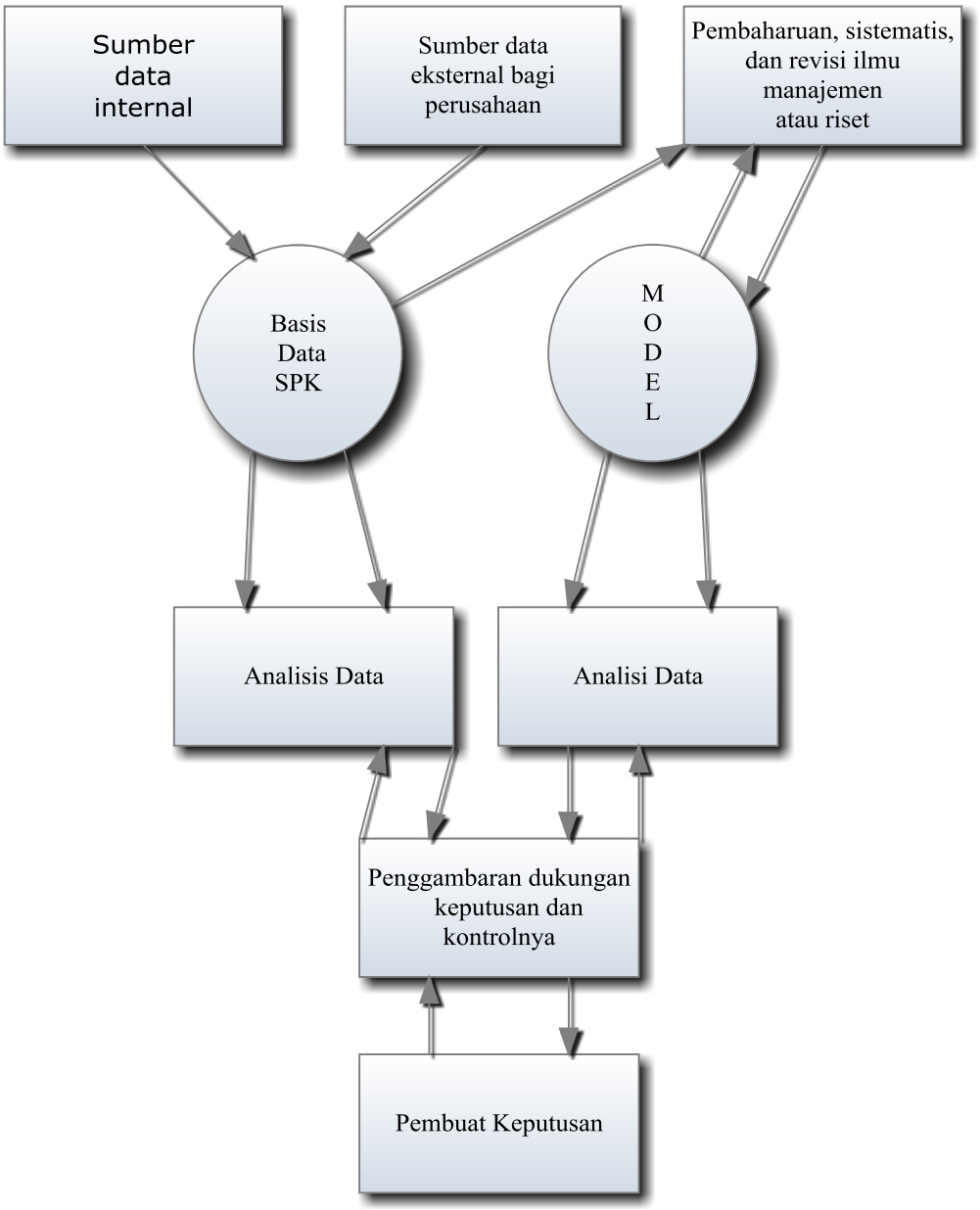
Pada dasarnya pengambilan keputusan adalah suatu pendekatan sistematis suatu masalah dengan pengumpulan fakta, penentuan yang matang dari alternatif yang dihadapi, dan pengambilan tindakan yang menurut perhitungan merupakan tindakan yang paling tepat. Pada sisi lain, pembuat keputusan kerap kali dihadapkan pada kerumitan dan lingkup pengambilan keputusan dengan data yang begitu banyak. Untuk kepentingan ini, sebagian besar pembuat keputusan dengan mempertimbangkan rasio manfaat/ biaya, dihadapkan pada suatu keharusan untuk mengandalkan seperangkat system yang mampu memecahkan masalah secara efisien dan efektif, yang kemudian disebut Sistem Pendukung Keputusan (SPK).

Pengertian yang hampir serupa, Sistem pendukung keputusan (decision support systems disingkat DSS) adalah bagian dari sistem informasi berbasis

komputer (termasuk sistem berbasis pengetahuan (manajemen pengetahuan)) yang dipakai untuk mendukung pengambilan keputusan dalam suatu organisasi atau perusahaan. Dapat juga dikatakan sebagai sistem komputer yang mengolah data menjadi informasi untuk mengambil keputusan dari masalah semi-terstruktur yang spesifik. Tujuan pembentukan SPK yang efektif adalah memanfaatkan keunggulan kedua unsur, yaitu manusia dan perangkat elektronik. Terlalu banyak menggunakan komputer akan menghasilkan pemecahan yang bersifat mekanis, reaksi yang tidak fleksibel, dan keputusan yang dangkal. Sedangkan terlalu banyak manusia akan memunculkan reaksi yang lamban, pemanfaatan data yang serba terbatas, dan kelambanan dalam mengkaji alternatif yang relevan. Guna membantu mempercepat dan mempermudah proses pengambilan keputusan, diperlukan suatu bentuk Sistem Pendukung Keputusan. Tujuannya adalah untuk membantu pengambilan keputusan memilih berbagai alternatif keputusan yang merupakan hasil pengolahan informasi yang diperoleh/ tersedia dengan menggunakan model pengambilan keputusan.

Sistem Pendukung Keputusan mempunyai karakteristik, yaitu kapabilitas interaktif, fleksibilitas, kemampuan mengintegrasikan model dan fleksibilitas output. Kapabilitas interaktif, SPK memberi pengambil keputusan akses cepat ke data dan informasi yang dibutuhkan. Fleksibilitas, SPK dapat menunjang para manajer pembuat keputusan di berbagai bidang fungsional (keuangan, operasi produksi, dan lain-lain). Kemampuan mengintegrasikan model, SPK memungkinkan para pembuat keputusan berintegrasi dengan model-model, termasuk memanipulasi sesuai dengan kebutuhan. dapat dilihat pada gambar

berikut,



Gambar II.12. Konfigurasi Sistem Pendukung Keputusan

(Sumber: Jurnal Teknologi, Volume 3 Nomor 2, Desember 2010, 145-153)

II.7. Subsistem Manajemen Basis Data

Ada beberapa perbedaan antara data base untuk SPK dan non-SPK. Pertama, sumber data untuk SPK lebih ”kaya” dari pada non-SPK dimana data harus berasal dari luar dan dari dalam karena proses pengambilan keputusan.

Perbedaan lain adalah proses pengambilan dan ekstraksi data dari sumber data yang sangat besar. SPK membutuhkan proses ekstraksi dan DBMS (*Database Management System*) yang dalam pengelolaannya harus cukup fleksibel untuk memungkinkan penambahan dan pengurangan secara cepat. Dalam hal ini, kemampuan yang dibutuhkan dari manajemen basis data dapat diringkas, sebagai berikut:

1. Terpadu, berarti bahwa berkas- berkas data yang ada pada basis data saling terkait, tetapi kemubaziran data tidak akan terjadi atau hanya terjadi sedikit sekali.
2. Berbagi Data, data dapat dipakai dari sejumlah pengguna. Lebih tegasnya lagi, sesuatu data dapat diakses oleh sejumlah pengguna dalam waktu bersamaan.
3. Kemubaziran data terkurangi, intrigitas data, independensi data, konsistensi data, berbagi data, sekuritas data, pengguna data lebih mudah.
4. Dalam lingkungan basis data, data lebih mudah digunakan. Pada beberapa DBMS tersedia fasilitas *query* yang memudahkan pengguna untuk memperoleh informasi.
5. Lapis pandangan merupakan lapis tertinggi pada abstraksidata, pada lapis ini pengguna hanya mengenal struktur data yang sederhana.(Abdul Kadir: 2003; 19)

II.7.1. Subsistem Manajemen Basis Model

Salah satu keunggulan SPK adalah kemampuan untuk mengintegrasikan akses data dan model-model keputusan. Hal ini dapat dilakukan dengan menambahkan model-model keputusan ke dalam sistem informasi yang menggunakan database sebagai mekanisme integrasi dan komunikasi di antara model-model. Karakteristik ini menyatukan kekuatan pencarian dan pelaporan data.

Salah satu persoalan yang berkaitan dengan model adalah bahwa penyusunan model seringkali terikat pada struktur model yang mengasumsikan adanya masukan yang benar dan cara keluaran yang tepat. Sementara itu, model cenderung tidak mencukupi karena adanya kesulitan dalam mengembangkan model yang terintegrasi untuk menangani sekumpulan keputusan yang saling bergantungan. Cara untuk menangani persoalan ini dengan menggunakan koleksi berbagai model yang terpisah, dimana setiap model digunakan untuk menangani bagian yang berbeda dari masalah yang dihadapi. Komunikasi antara berbagai model digunakan untuk menangani bagian yang berbeda dari masalah tersebut. Komunikasi antara berbagai model yang saling berhubungan diserahkan kepada pengambil keputusan sebagai proses intelektual dan manual.

Salah satu pandangan yang lebih optimis, berharap untuk bisa menambahkan model-model ke dalam sistem informasi dengan database sebagai mekanisme integrasi dan komunikasi di antara mereka.

Kemampuan yang dimiliki subsistem basis model meliputi:

1. Kemampuan untuk menciptakan model-model baru secara cepat dan mudah.

2. Kemampuan untuk mengakses dan mengintegrasikan model-model keputusan.
3. Kemampuan untuk mengelola basis model dengan fungsi manajemen yang analog dan manajemen database (seperti mekanisme untuk menyimpan, membuat dialog, menghubungkan, dan mengakses model).

II.8. Unified Modeling Language (*UML*)

UML singkatan dari *Unified Modeling Language* yang berarti bahasa pemodelan standart. Mengatakan sebagai bahasa, berarti *UML* memiliki sintaks dan semantik. Ketika kita membuat model menggunakan konsep *UML* ada aturan-aturan yang harus diikuti. Bagaimana elemen pada model-model yang kita buat berhubungan satu dengan yang lainnya harus mengikuti standart yang ada. *UML* bukan hanya sekedar diagram, tetapi juga menceritakan konteksnya. Ketika pelanggan memesan sesuatu dari sistem, bagaimana transaksinya ? bagaimana mengatasi error yang terjadi ? Bagaimana keadaan terhada sistem yang kita buat ? Dan sebagainya dapat dijawab dengan *UML*.

UML diaplikasikan untuk maksud tertentu, biasanya antara lain untuk:

1. Merancang perangkat lunak.
2. Sarana komunikasi antara perangkat lunak proses bisnis.
3. Menjabarkan sistem sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem.
4. Mendokumentasi sistem yang ada, proses, proses dan organisasinya.

UML telah diaplikasikan dalam bidang investasi perbankan, lembaga kesehatan, departemen pertahanan, sistem terditribusi, sistem pendukung alat kerja, retail, sales dan supplier.

Block pembangun utama *UML* adalah diagram. Beberapa diagram ada yang rinci (jenis *timing diagram*) dan lainnya ada yang bersifat umum (misal diagram kelas). Para pengembang sistem beriorientasiobjek menggunakanbahasa model untuk menggambarkan, membangun dan mendokumentasikan sistem yang mereka rancang. *UML* memungkinkan para anggota team untuk bekerja sama dengan bahasa model yang sama dalam mengaplikasikan beragam sistem. Intinya, *UML* merupakan alat komunikasi yang konsisten dalam mensuport para pengembang sistem saat ini. Sebagai perancang sistem, mau tidak mau pasti akan menjumpai *UML*, baik kita sendiri yang membuat atau sekedar membaca diagram *UML* buatan orang lain (Prabowo Pudjo Widodo Helrlawa, Herlawati: 2011; 7).

II.8.1. Diagram- Diagram *UML*

Beberapa literatur bahwa *UML* menyediakan sembilan jenis diagram, yang lain menyebutkan delapan, karena ada beberapa diagram yang digabung, misalnya diagram komunikasi, diagram urutan dan diagram perwaktuan digabung menjadi diagram interaksi. Namun demikian model- model itu dapat dikelompokkan berdasarkan sifatnya yaitu statis atau dinamis. Jenis diagram itu antara lain:

1. Diagram Kelas. Bersifat statis. Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka- antarmuka, kolaborasi- kolaborasi, serta relasi- relasi.

Diagram ini umum dijumpai pada pemodelan sistem berorientasi objek. Meskipun bersifat statis, sering pula diagram kelas memuat kelas- kelas aktif.

2. Diagram Paket (*Package Diagram*). Bersifat statis. Diagram ini memperlihatkan kumpulan kelas- kelas, merupakan bagian diagram komponen.
3. Diagram *Use-Case*. Bersifat statis. Diagram ini memperlihatkan himpunan *Use-Case* dan aktor- aktor (suatu jenis khusus dari kelas). Diagram ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku suatu sistem yang dibutuhkan serta diharapkan pengguna.
4. Diagram Interaksi dan *Sequence* (urutan). Bersifat dinamis. Diagram urutan adalah diagram interaksi yang menekannkan pada pengiriman pesan dalam suatu waktu tertentu.
5. Diagram Komunikasi (*Communication Diagram*). Bersifat dinamis. Diagram sebagai pengganti diagram kolaborasi *UML* yang menekannkan organisasi struktural dari objek- objek yang menerima serta mengirim pesan.
6. Diagram *Statechart* (*Statechart Diagram*). Bersifat dinamis. Diagram status memperlihatkan keadaan- keadaan pada sistem, memuat status (*state*), transisi, kejadian serta aktifitas. Diagram ini terutama penting untuk memperlihatkan sifat dinamis dari Antarmuka (*interface*), kelas, kolaborasi dan terutama penting pada pemodelan sitem- sistem yang reaktif.
7. Diagram Aktivitas (*Activity Diagram*). Bersifat dinamis. Diagram aktivitas adalah tipe khusus dari diagram status yang memperlihatkan aliran dari suatu aktivitas ke aktivitas lainnya dalam suatu sitem. Diagram ini terutama penting

dalam pemodelan fungsi- fungsi suatu sistem dan memberi tekanan pada aliran kendali antar objek.

8. Diagram Komponen (*Component Diagram*). Bersifat statis. Diagram komponen ini memperlihatkan organisai serta ketergantungan sistem/ perangkat lunak pada komponen- komponen yang telah ada sebelumnya. Diagram ini berhubungan dengan diagram kelas dimana komponen secara tipikal dipetakan kedalan satu atau lebih kelas- kelas, Antarmuka- Antarmukaserta Kolaborasi-kolaborasi.
9. Diagram *Deployment* (*Deployment Diagram*). Bersifat statis. Diagram ini memperlihatkan konfigurasi pada saat aplikasi dijalankan (*run-tine*). Memuat simpul- simpul beserta komponen- komponen yang ada di dalamnya. Diagram *deployment* berhubungan erat dengan diagram komponen dimana diagram ini memuat satu atau lebih komponen-komponen. Diagram ini sangat berguna saat aplikasi kita berlaku sebagai aplikasi yang dijalankan pada banyak mesin (*distributeed computing).*

Kesembilan diagram ini tidak mutlak harus digunakan dalam pengembangan perangkat lunak, semuanya dibuat sesuai dengan kebutuhan. Pada UML dimungkinkan kita menggunakan diagram-diagram lainnya (misalnya *data flow diagram, entity relationship diagram* dan sebagainya).

II.8.2. Diagram *Use-Case* (*Use-Case* Diagram)

UML menyediakan serangkaian gambar dan diagram yang sangat baik. Beberapa diagram memfokuskan diri pada tangguhan teori *object- oriented* dan

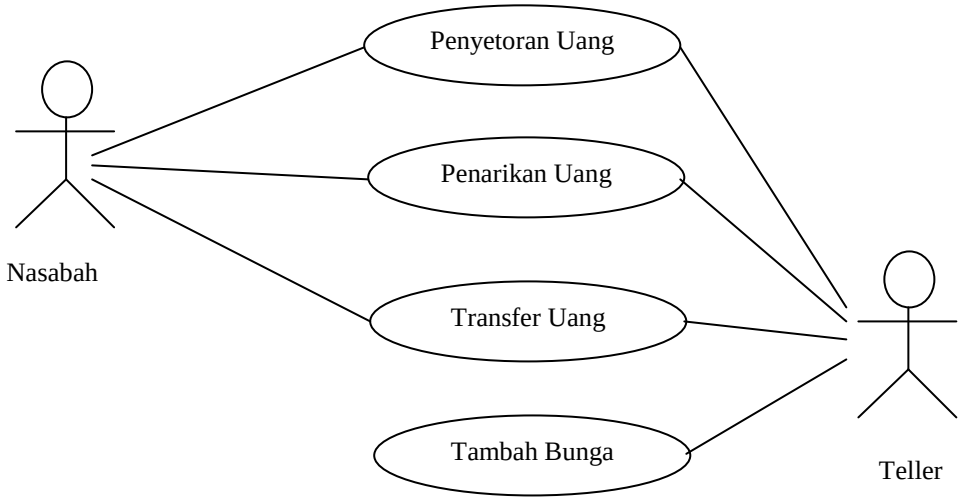
sebagian lagi memfokuskan pada detil rancangan dan konstruksi. Semuanya dimaksudkan sebagai sarana komunikasi antan team programer maupun dengan pengguna. Sistem yang kita buat tidak selalu menggambarkan aktivitas internal, hubungan dengan suplier dan pelanggan yang bersipat esternal harus diperhatikan.

Salah satu kontributor terhadap diagram *use case* dalam UML adalah Ivar Jacobsen. *Use case* menggambarkan external view dari sistem yang akan kita buat modelnya. Mengatakan bahwamodel *use case* dapat dijabarkan dalam diagram *use case*, tetapi yang perlu diingat, diagram tidak identik dengan model karena model lebih luas dari diagram.

Komponen pembentuk diagram *use case* adalah:

1. Aktor (actor),menggambarkan pihak- pihak yang berperan dalam sistem
2. *Use-case*, aktivitas/ sarana yang disiapkan oleh bisnis/sistem.
3. Hubungan (*link*),aktor mana saja yang terlibat didalam *use-case* ini.

Gambar II.11 dibawah ini merupakan salah satu contoh bentuk diagram *use-case.*



Gambar II.13 Diagram Use Case

(Sumber: Prabowo Pudjo Widodo Helrlawa, Herlawati: 2011; 7)

II.9. Sejarah Java

Java telah hadi dalam dunia pemrograman selama satu dekate lebih. Sudah ada beberapa bahasa pemrograman lainnya yang berusaha menyamai bahkan menggantikan kedudukan java sebagai pemrograman yang pertama kali memperkenalkan pemrograman lintas platform secara independen tidak tergantung pada sebuah mesin. Mendukung konsep OPP (*Object Oriented Programming*) secara total. Maksudnya adalah sejak awal mula java diciptakan, java memang dibuat untuk mendukung konsep OPP tersebut. Oleh karena itu, struktur bahasa pemrograman Java harus memiliki sebuah class utama dan sebuah method utama (*main method*).

Bahasa pemrograman Java dimulai dari sebuah tim pengembang *software* dari Sun Microsystem yang dipimpin oleh James Gosling dan Patrick Naughton. Pada tahun 1991, Sun Microsystem mengembangkan sebuah bahasa pemrograman yang berukuran kecil untuk diimplementasikan pada alat elektronik rumah tangga seperti swichbox TV kabel. Berhubung alat tersebut tidak memiliki banyak memori, maka bahasayang digunakan harus sangat kecil dan menghasilkan kode yang kecil pula. Permasalahan lainnya adalah alatnya adalah alat- alat tersebut memiliki CPU yang berbeda- beda karena dibuat oleh manufaktur yang berbeda. Jadi sangat diharuskan bahasa pemrograman tersebut tidak terikat pada sebuah arsitektur mesin tertentu saja

Oleh karena adanya keharusan sebuah bahasa pemrograman yang kecil, menghasilkan kode yang kecil pula dah harus platform independen (tidak terikat pada platform) membuat tim pada proyek tersebut terinfirasi olej ide pemrograman

yang sama yang telah ditemukan oleh Niklaus Wirth, penemu Pascal. Jadi, penemu bahasa Pascal juga memiliki pemikiran tentang sebuah *software* bahasa pemrograman yang portabel dan tidak tergantung pada sebuah platform atau mesin. Bahasa pemrograman komersial yang disebut UCSD Pascal tersebut menghasilkan kode intermediate yang diperuntukkan bagi sebuah mesin virtual. Jadi, kode asli dari bahasa pemrograman tersebut tidak tergantung pada mesin ataupun platform sistem operasi UCSD Pascal menghasilkan intermediate *code* yang selanjutnya adakan di kompilasi atau diterjemahkan oleh mesin virtual ke kode mesin dimana kode tersebut dijalankan.

Pada tahun- tahun tersebut perkembangan internet sangat pesat. Namun saat itu *browser* juga masih jarang ditemui, pada tahun 1994 kebanyakan orang menggunakan Mosaic, yaitu sebuah *browser* nonkomersial yang dibuat oleh Marc Andreessen pada tahun 1993 di *supercomputing center* Universitas Illionis. Pada pertengahan tahun 1994 para pengambang java menyadari bahwa mereka dapat saja membangun sebuah browser yang lebih fleksibel daripada yang lainnya. Selanjutnya *browser* yang dikerjakan oleh Patrick Naugton dan Jonathan Payne. Tujuan utama dari pembuatan *browser* tersebut adalah tidak lain untuk mempromosikan bahasa java dan memamerkan kekuatannya. Java juga memiliki kekuatan pada aplikasi yang disebut applet yang juga berhubungan dengan *browser*.

Booming bahasa Java dimulai pada tahun 1995 ketika *Netscape* memutuskan untuk menggunakan Java pada *web browser*-nya, yaitu *Netscape Navigator* pada januari 1996. Hal ini kemudian diikuti oelh raksasa- raksasa

software seperti IBM, Symantec, Inprise, dan masih banyak yang lainnya termasuk Microsoft dengan Internet Explorer-nya. (Sumber: Andi; 2010: 1-3)

II.9.1. Keunggulan Java

Berikut adalah keunggulan-keunggulan Java dibandingkan dengan bahasa pemrograman lain.

1. Berbasis GUI

   Kita bisa membuat tampilan program berbasis grafik (*Graphical User Interface*/GUI) untuk memudahkan pemakai berinteraksi dengan program.

2. Berorientasi Objek

   Konsep pemrograman berorientasi objek tak lain dirancang agar kita dapat memandang pemrograman sebagai suatu kehidupan nyata. Ini membuat pengembang software menjadi lebih mudah karena kita seolah-olah berhubungan dengan kehidupan nyata. Java merupakan salah satu bahasa yang memiliki dukungan penuh terhadap konsep pemrograman berorientasi objek ini.

3. Aplikasi Web

   Saat ini web merupakan sarana yang tidak dipisahkan dari IT. Mungkin ini disebabkan fitur web sebagai sarana komunikasi dihubungkan dengan konsep bahwa manusia adalah makhluk sosial yang perlu berkomunikasi. Java merupakan bahasa pemrograman yang memiliki dukungan sangat baik terhadap aplikasi web. Memang pada awalnya Java dilahirkan sebagai solusi untuk menjawab kebutuhan bahasa pengembangan yang mendukung aplikasi

berbasis jaringan. Diantara teknologi yang mendukung aplikasi web adalah Applet, JSP (untuk aplikasi web berbasis server), CORBA (untuk aplikasi terdistribusi), dan lain-lain.

4. Multi platform

Java berbeda dengan bahasa pemrograman lain misalnya C, C++ atau Pascal. Jika Anda membuat program dengan bahasa Pascal misalnya, maka program yang anda compile hanya bisa berjalan di satu platform saja. Jika di-compile di Linux, maka hanya bisa dijalankan di Linux. Jika Anda di-compile di *Windows*, maka program hanya bisa dijalankan di windows saja. Jika Anda membuat program untuk berjalan di atas Windows, kemudian suatu waktu Anda ingin menjalankannya di Linux, maka Anda memerlukan usaha yang berat untuk menyesuaikan kode program tersebut. Dengan Java, hal tersebut tidak akan terjadi. Sekali Anda membangun program, maka akan bisa dijalankan di berbagai platform komputer, dengan syarat JVM telah ter-install pada platform tersebut. Jika Anda membuat program dengan Java di atas Linux, maka akan bisa dijalankan juga di *Windows* dan Macintosh. Selain itu, Java bisa juga digunakan untuk membuat program yang berjalan di atas mobile device (dengan J2ME), PDA, embede system, dan lain-lain.

II.10. Program Java dengan Netbean IDE

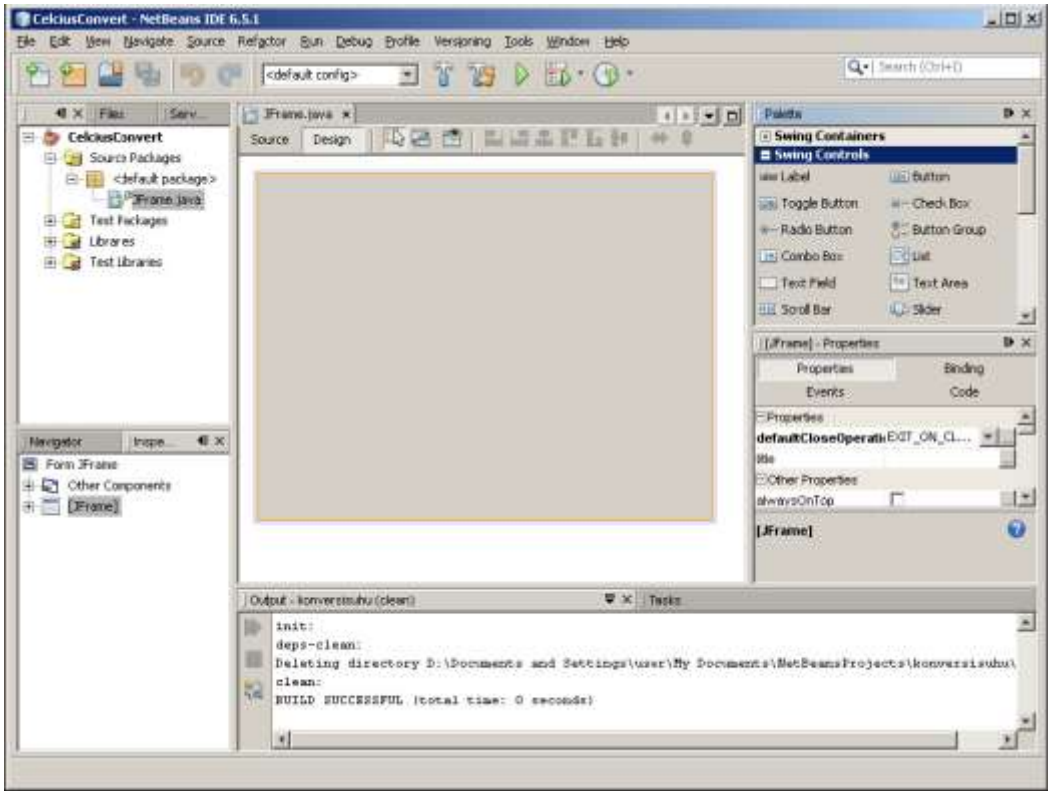
1. Sekilas Tentang Netbean

The NetBeans IDE adalah sebuah lingkungan pengembangan – sebuah tools untuk programmer menulis, mengompilasi, mencari kesalahan dan menyebarkan

program. Netbeans IDE ditulis dalam Java – namun dapat mendukung bahasa pemrograman lain. Terdapat banyak modul untuk memperluas Netbeans IDE. Netbeans IDE adalah sebuah produk bebas dengan tanpa batasan bagaimana digunakan. (dikutip dari : www.netbean.org) NetBeans mengacu pada dua hal, yakni platform untuk pengembangan aplikasi desktop java, dan sebuah *Integrated Development Environment* (IDE) yang dibangun menggunakan platform NetBeans. Platform NetBeans memungkinkan aplikasi dibangun dari sekumpulan komponen perangkat lunak moduler yang disebut 'modul'. Sebuah modul adalah suatu arsip Java (*Java archive*) yang memuat kelas-kelas Java untuk berinteraksi dengan NetBeans Open API dan file manifestasi yang mengidentifikasinya sebagai modul. Aplikasi yang dibangun dengan modul-modul dapat dikembangkan dengan menambahkan modul-modul baru. Karena modul dapat dikembangkan secara independen, aplikasi berbasis platform NetBeans dapat dengan mudah dikembangkan oleh pihak ketiga secara mudah dan powerful. Pengembangan NetBeans diawali dari Xelfi, sebuah proyek mahasiswa tahun 1997 di bawah bimbingan Fakultas Matematika dan Fisika Universitas Charles, Praha. Sebuah perusahaan kemudian dibentuk untuk proyek tersebut dan menghasilkan versi komersial NetBeans IDE hingga kemudian dibeli oleh Sun Microsystem pada tahun 1999. Sun kemudian menjadikan NetBeans *open source* pada bulan Juni tahun 2000. Sejak itu komunitas NetBeans terus berkembang.

2. Pengenalan IDE (*Integrated Development Environment*) Netbean

Lingkungan Pengembangan yang terintegrasi pada netbean IDE, memudahkan pengguna untuk membuat beragam aplikasi dengan mudah.



Gambar II.14 Tampilan IDE Netbean

(Sumber: Andi; 2010: 130)