BAB II

TINJAUAN PUSTAKA

II.1. Sistem Pakar (*Expert System*)

Sistem pakar adalah suatu sistem yang dirancang untuk dapat menirukan keahlian seorang pakar dalam menjawab pertanyaan dan memecahkan suatu masalah. Sistem pakar akan memeberikan pemecahan suatu masalah yang didapat dari dialog dengan pengguna. Dengan bantuan sistem pakar seseorang yang bukan pakar/ahli dapat menjawab pertanyaan, menyelesaikan masalah serta mengambil keputusan yang biasanya dilakukan oleh seorang pakar (T. Sutojo;2010;13).

II.1.1. Konsep Dasar Sistem Pakar

Konsep dasar sistem pakar meliputi enam hal berikut ini :

1. **Kepakaran (*Expertise*)**

   Kepakaran merupakan suatu pengetahuan yang diperoleh dari pelatihan, membaca, dan pengalaman. Kepakaran ialah yang memungkinkan para ahli dapat mengambil keputusan lebih cepat dan lebih baik daripada seseorang yang bukan pakar. Kepakaran itu sendiri meliputi pengetahuan tentang :

   a. Fakta-fakta tentang bidang permasalahan tertentu,
   b. Teori-teori tentang bidang permasalahn tertentu,
   c. Aturan-aturan dan prosedur-prosedur menurut bidang permasalahan umumnya,
   d. Aturan *heuristic* yang harus dikerjakan dalam situasi tertentu,

e. Strategi global untuk memecahkan permasalahan, dan

f. Pengetahuan tentang pengetahuan (*meta knowledge*).

2. Pakar (*Expert*)

Pakar adalah seseorang yang mempunyai pengetahuan, pengalaman, dan metode khusus, serta mampu menerapkannya untuk memecahkan masalah atau member nasihat.

3. Pemindahan Kepakaran (*Transfering Expertise*)

Tujuan dari sistem pakar adalah memindahkan kepakaran dari seorang pakar ke dalam computer, kemudian ditransfer kepada orang lain yang bukan pakar. Proses ini melibatkan empat kegiatan, yaitu :

a. Akuisisi pengetahuan (dari pakar atau sumber lain),

b. Representasi pengetahuan (pada komputer),

c. Inferensi pengetahuan, dan

d. Pemindahan pengetahuan ke pengguna.

4. Inferensi (*Inferencing*)

Inferensi adalah sebuah prosedur (program) yang mempunyai kemampuan dalam melakukan penalaran. Inferensi ditampilkan pada suatu komponen yang disebut mesin. Inferensi yang mencakup prosedur-prosedur mengenai pemecahan masalah.

5. Aturan-aturan (*Rule*)

Kebanyakan software sistem pakar komersial adalah sistem yang berbasis *rule* (*rule-based systems*), yaitu pengetahuan yang disimpan terutama dalam bentuk *rule* , sebagai prosedur pemecahan masalah.

6. Kemampuan menjelaskan (*Explanation Capability*)

Fasilitas lain dari sistem pakar adalah kemampuannya untuk menjelaskan saran atau rekomendasi yang diberikannya. Penjelasan dalam subsistem yang disebut subsistem penjelasan (*explanation*). Bagian dari sistem ini memungkinkan sistem untuk memeriksa penalaran yang dibuatnya sendiri dan menjelaskan operasi-operasinya (T.Sutojo;2010;163-165).

Karakteristik dan kemampuan yang dimiliki oleh sistem pakar berbeda dengan sistem konvensional. Perbedaan ini ditunjukkan pada Tabel II.1 yang ada dibawah ini.

Tabel II.1 Perbandingan antara Sistem Konvensional dengan Sistem Pakar

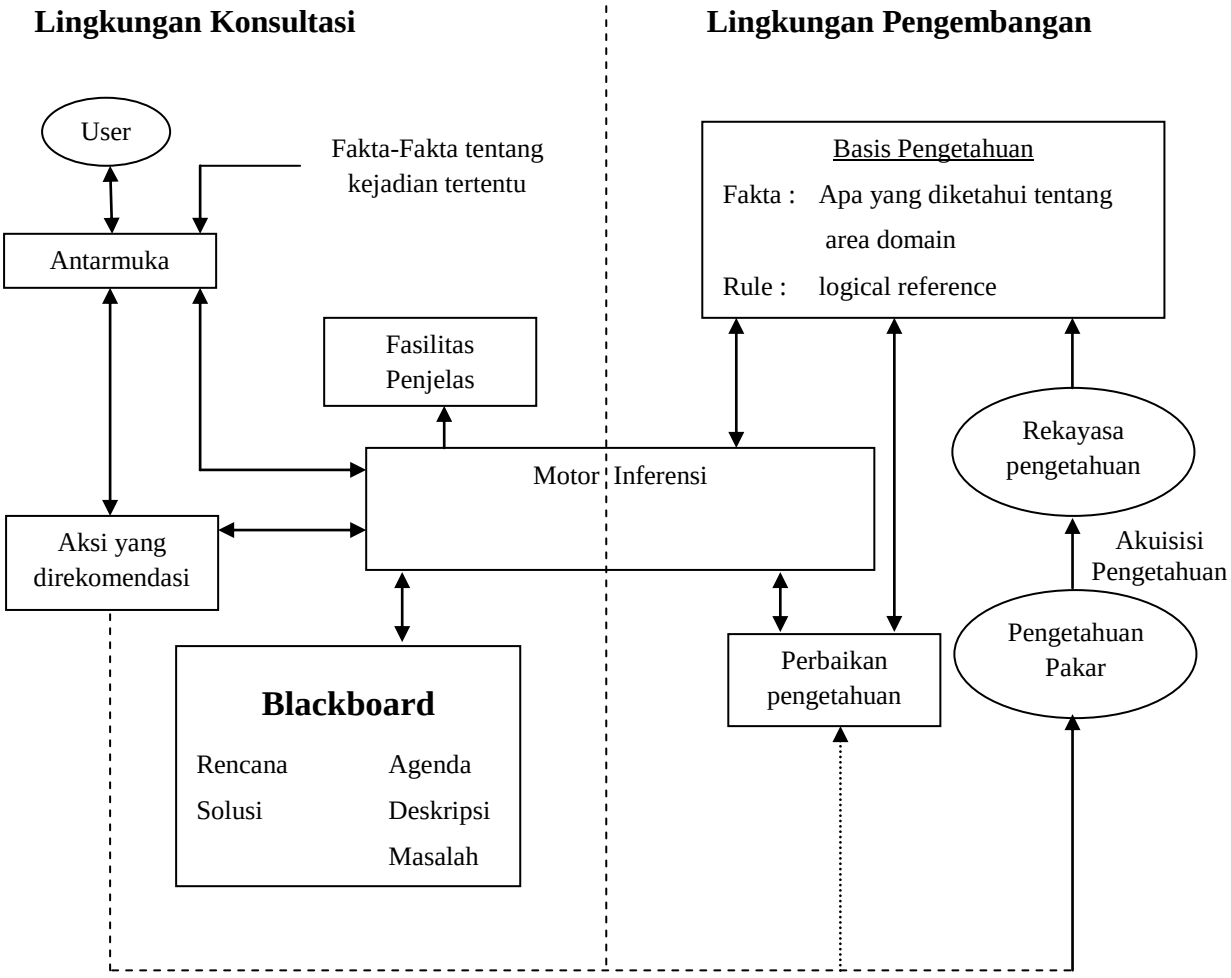
Sistem Konvensional	Sistem Pakar
Informasi dan pemprosesannya biasanya digabungkan dalam satu program.	Basis pengetahuan diposahkan secara jelas dengan mekanisme inferensi.
Program tidak membuat kesalah (yang membuat kesalahan: Pemrogram atau Pengguna).	Program dapat berbuat kesalahan.
Biasanya tidak menjelaskan mengapa data masukan diperlakukan atau bagaimana output dihasilkan.	Penjelasan merupakan bagian terpenting dari semua sistem pakar.
Perubahan program sangat menyulitkan.	Perubahan dalam aturan-aturan mudah untuk dilakukan.
Sistem hanya bisa beroperasi setelah lengkap atau selesai.	Sistem dapat beroperasi hanya dengan aturan-aturan yang sedikit (sebagai prototipe awal).
Eksekusi dilakukan berdasarkan langkah demi langkah (*algoritmatic*).	Eksekusi dilakukan dengan menggunkaan *heuristic* dan logika pada seluruh basi pengetahuan.
Perlu informasi lengkap agar bisa beroperasi.	Dapat beroperasi dengan infomasi yang tidak lengkap atau mengandung ketidakpastian.
Menaipulasi efektif dari basis data yang besar.	Manipulasi efektif dari basis pengetahuan yang besar.
Menggunkan data.	Menggunakan pengetahuan.
Tujuan utama efisiensi.	Tujuan utama efektivitas.
Mudah berurusan dengan data	Mudah berurusan dengan data

kuantutatif.	kulaitatif.
Menagkap, menambah dan mendistribusikan akses ke data numerik atau informasi.	Menangkap, menambah dan mendistribusikan akses ke pertimbangan dan pengetahuan.

Sumber: (T. Sutojo;2011:165)

II.1.2. Struktur Sistem Pakar

Ada dua bagian penting dari Sistem Pakar, yaitu lingkungan pengembangan (*development environment*) dan lingkungan konsultasi (*consultation environment*). Lingkungan pengembangan digunakan oleh pembuat sistem pakar untuk membangun komponen-komponennya dan memperkenalkan pengetahuan ke dalam *knowledge base* (basis pengetahuan). Lingkungan konsultasi digunakan oleh pengguna untuk berkonsultasi sehingga pengguna mendapatkan pengetahuan dan nasihat dari Sistem Pakar layaknya berkonsultasi dengan seorang pakar. Gambar II.1 menunjukkan komponen-komponen yang penting dalam sebuah sistem pakar



Gambar II.1 Komponen-Komponen yang penting dalam sebuah sistem pakar

Sumber : (T.Sutojo;2011;167)

Keterangan gambar :

1. Akuisisi pengetahuan

   Subsistem ini digunakan untuk memasukkan pengetahuan dari seorang pakar dengan cara merekayasa pengetahuan agar bisa diproses oleh komputer dan menaruhnya ke dalam basis pengetahuan dengan format tertentu (dalam bentuk representasi pengetahuan). Sumber-sumber pengetahuan bias diperoleh

dari pakar, buku, dokumen multimedia, basis data, laporan riset khusus, dan informasi yang terdapat di Web

2. Basis pengetahuan (*Knowledge Base*)

Basis pengetahuan mengandung pengetahuan yang diperlukan untuk memahami, memformulasikan, dan menyelesaikan masalah. Basis pengetahuan terdiri dari dua elemen dasar, yaitu :

a. Fakta, misalnya situasi, kondisi, atau permasalahan yang ada.
b. Rule (aturan), untuk mengarahkan penggunaan pengetahuan dalam memecahkan masalah.

3. Motor inferensi (*Inference Engine*)

Motor atau mesin inferensi adalah sebuah program yang berfungsi untuk memandu proses penalaran terhadap suatu kondisi berdasarkan pada basis pengetahuan yang ada, memanipulasi dan mengarahkan kaidah, model, dan fakta yang disimpan dalam basis pengetahuan untuk mencapai solusi atau kesimpulan. Dalam prosesnya, mesin inferensi menggunakan strategi pengendalian, yaitu strategi yang berfungsi sebagai panduan arah dalam melakukan proses penalaran. Ada tiga teknik pengendalian yang digunakan, yaitu *forward chaining*, *backward chaining*, dan gabungan dari kedua teknik tersebut.

4. Daerah Kerja (*Blackboard*)

Untuk merekam hasil sementara yang akan dijadikan keputusan dan untuk menjelaskan sebuah masalah yang sedang terjadi, sistem pakar membutuhkan

Blackboard, yaitu area pada memori yang berfungsi sebagai basis data. Tiga tipe keputusan yang dapat direkam pada *blackboard*, yaitu :

a. Rencana : bagaimana menghadapi masalah
b. Agenda : aksi-aksi potensial yang sedang menunggu untuk dieksekusi
c. Solusi : calon aksi yang akan dibangkitkan

5. Antarmuka Pengguna (*Interface*)

Digunakan sebagai media komunikasi antara pengguna dengan sistem pakar. Komunikasi ini paling bagus bila disajikan dalam bahasa alami (*natural language*) dan dilengkapi dengan grafik, menu, dan formulir elektronik. Pada bagian ini akan terjadi dialog antara sistem pakar dengan pengguna.

6. Subsistem Penjelasan (*Explanation Subsystem / Justifier*)

Berfungsi member penjelasan kepada penggunabagaimana suatu kesimpulan dapat diambil. Kemampuan seperti ini sangat penting bag pengguna untuk mengetahui proses pemindahan keahlian pakar maupun dalam pemecahan masalah.

7. Sistem Perbaikan Pengetahuan (*Knowledge Refining System*)

Kemampuan memperbaiki pengetahuan dari seorang pakar diperlukan untuk menganalisis pengetahuan, belajar dari kesalahan masa lalu, kemudian memperbaiki pengetahuannya sehinnga dapat dipakai pada masa mendatang. Kemampuan evaluasi diri seperti itu diperlukan oleh program agar dapat mmenganalisis alasan-alasan kesuksesan dan kegagalannya dalam mengambil kesimpulan. Dengan cara ini basis pengetahuan yang lebih baik dan penalaran yang lebih efektif akan dihasilkan.

8. Pengguna (*User*).

Pada umumnya pengguna sistem pakar bukanlah seorang pakar (*non-expert*) yang membutuhkan solusi, saran, atau pelatihan (*training*) dari berbagai permasalahn yang ada. (T.Sutojo;2011;167-169)

II.2. Penyakit Cacar

Dalam tahun-tahun 1790-an telah dibuktikan bahwa infeksi karena cowpox dapat memberikan kekebalan terhadap penyakit cacar (smallpox), tetapi baru 200 tahun kemudian prinsip ini diterima dan diterapkan diseluruh dunia sehingga penyakit cacar dapat dibasmi dari seluruh dunia (tahun 1978 sudah tidak ada lagi kasus cacar). Program pembasmian cacar ini dikoordinasikan oleh WHO dan dimulai pada tahun 1967 (suatu program pembasmian 10 tahun).

Epidemiologi terutama berperan dalam hal menentukan distribusi kasus, dan model mekanisma dan derajat penyebaran , dengan jalan pemetaan meletupnya penyakit tersebut , dan melakukan evaluation program penanggulangan. Faktor-faktor yang menunjang keberhasilan pembasman cacar adalah: kemauan polotik, tujuan yang jelas, jadwal yang tepat, staf yang terlatih, dan strategi yang luwes, disamping itu juga terdapatnya vaksin yang tahan terhadap panas dan efektif (Ridwan amirudin, dkk;2011;22)

1. Cacar Air (*Varicella Simple / Chickenpox*)

Cacar air adalah penyakit yang amat menular yang disebabkan oleh virus varisela-zoster. Cacar air biasanya mulai dengan gejala-gejala mirip masuk angin, seperti pilek, demam ringan, batuk dan rasa capek, diikuti oleh

bintik-bintik merah yang biasanya terlihat pada batang tubuh dan wajah serta menyebar ke seluruh tubuh. Bercak-bercak merah mulai sebagai bintik kecil-kecil merah yang dengan cepat berubah menjadi lepuhan-lepuhan. Cacar air tersebar lewat batuk dan bersin dan lewat kontak langsung dengan cairan dalam lepuhan bintik-bintik merah.

Pada anak-anak sehat, cacar air biasanya merupakan penyakit ringan yang berlangsung sekitar 5–10 hari. Bintik-bintik merah cacar air mungkin amat gatal dan menggaruknya dapat mengakibatkan infeksi-infeksi bakterial pada bercak-bercak tersebut. Anak-anak dengan kondisi medis lain berisiko kena komplikasi yang mengancam hidup seperti pneumonia atau inflamasi otak (ensefalitis). Jika seorang ibu mengidap cacar air selama kehamilannya, ada kemungkinan kecil (kurang dari 2%) kerusakan pada bayi yang belum lahir kecuali menjelang kelahiran. Serangan infeksi cacar air pada ibu dalam 5 hari sebelum sampai 2 hari setelah melahirkan, akan mengakibatkan infeksi parah pada sebanyak sepertiga jumlah bayi yang baru saja dilahirkan. Orang-orang dewasa cenderung lebih parah gejala penyakit cacar airnya daripada anak-anak dan berkemungkinan besar kena komplikasi. (Commonwealth of Australia;2005;41).

2. Cacar Ular (*Herpes Zoster*)

Herpes zoster adalah penyakit yang disebabkan oleh infeksi virus varisela-zoster (VVZ) yang menyerang kulit dan mukosa, infeksi ini merupakan reaktivasi virus yang terjadi setelah infeksi primer. Infeksi primer dengan virus varisela zoster menimbulkan varisela (cacar air). Virus

membentuk infeksi laten di ganglia dorsal sehingga menyebabkan terjadinya herpes zoster.

Virus varisela zoster merupakan rantai ganda DNA (*deoxyribonucleic acid*) atau biasa disebut asam deoksiribonukleat yang termasuk dalam anggota famili virus herpes yang tergolong virus neuropatik atau neurodermatotropik. Reaktivasi virus varisela zoster dapat dipicu oleh berbagai faktor seperti antara lain pembedahan, penyinaran, lanjut usia, dan keadaan tubuh yang lemah meliputi malnutrisi, seseorang yang dalam pengobatan imunosupresan jangka panjang, atau menderita penyakit sistemik (Sahriani HR,dkk;2012;2).

II.3. Logika Fuzzy (*Fuzzy Logic*)

Logika fuzzy adalah metodologi sistem control pemecahan masalah yang cocok untuk diimplementasikan pada sistem, mulai dari sistem yang sederhana, sistem kecil, embedded system, jaringan PC, multi channel atau workstation berbasis akuisisi data, dan sistem control.

Beberapa alasan yang dapat diutarakan mengapa kita menggunakan logika fuzzy di antaranya adalah mudah dimengerti, memiiki toleransi terhadap data-data yang tidak tepat, mampu memodelkan fungsi-fungsi non-linear yang sangat kompleks, dapat membangun dan mengaplikasikan pengalaman-pengalaman para pakar secara langsung tanpa harus melalui proses pelatihan, dapat bekerja sama dengan teknik-teknik kendali secara konvensional, dan didasarkan pada bahasa alami.

Untuk memahami logika fuzzy, perlu dahulu memahami konsep variable linguistic, himpunan fuzzy, semesta pembicaraan, dan domain himpunan fuzzy. Fuzzy keanggotaan adalah grafik yang mewakili besar dari derajat keanggotaan masing-masing variabel input yang berada dalam interval 0 dan 1.

Derajat keanggotaan sebuah variabel x dilambangkan dengan symbol μ(x). ada beberapa fungsi keanggotaan yang sering digunakan,diantaranya adalah grafik keanggotaan kurva linear, segitiga, trapesium, bentuk baku, kurva –s (*sigmoid*), dan bentuk lonceng. (T.Sutjo;2010;276-277)

1. Desain Sistem Pakar Fuzzy

Pembuatan desain sistem yang digunakan sebagai contoh kasus adalah sistem pakar fuzzy untuk diagnose kanker prostat. Data yang diperlukan adalah antigen spesifik prostat (*Prostate Spesific Antigen /PSA*), umur pasien (*Age*), dan volume prostat (*Prostate Volume /PV*) sebagai parameter masukan dan skala resiko kanker prostat (*Prostate Cancer Risk /PCR*) sebagai keluaran.

Untuk proses fuzzifikasi beberapa parameter tersebut digunakan variabel linguistik *very small* (*VS*), *small* (*S*), *middle* (*M*), *high* (*H*), *very high* (*VH*), *very low* (*VL*), dan *Low* (*L*), serta menggunakan metode Mamdani maxmin dalam proses inferensi-nya. Faktor-faktor yang digunakan dalam sistem adalah : PSA (ng/ml), Umur (tahun), PV (ml), dan PCR (%). Untuk mendukung proses inferensi berbasis aturan maka dikembangkan 80 aturan fuzzy, seperti dalam tabel berikut.

Tabel II.2 Aturan (*rule*) yang dipakai

Nomor Rule	PSA	Umur Pasien	PV	PCR
Rule 1	VL	Very Young	VS	VL
Rule 2	VL	Very Young	VS	L
Rule 3	VL	…	…	
…	…			
Rule 44	VL	MA	H	VL
…				
Rule 77	VH	Old	VS	H

Sumber : (Nurul Hidayat, M. Munawar Yusro;2007;3)

Sebagai contoh, Aturan 1, Aturan 43, dan 77 dapat diinterpretasikan sebagai berikut :

Aturan 1 :

If PSA = VL and Umur = very young and PV = VS, then PCR= VL, atau jika dalam kalimat : Jika PSA Pasien adalah Very Low dan Pasien very young, dan PV pasien adalah Very Small, maka resiko kanker prostatnya adalah Very Low.

Aturan 43:

If PSA = VL and Umur = Middle and PV = H, then PCR= VL, atau jika dalam kalimat : Jika PSA Pasien adalah Very Low dan Umur Pasien Middle, dan PV pasien adalah Very Small, maka resiko kanker prostatnya adalah Very Low.

Aturan 77:

If PSA = VH and Umur = Old and PV = VS, then PCR= VH, atau jika dalam kalimat : Jika PSA Pasien adalah Very High dan Umur Pasien Old, dan PV pasien adalah Very Small, maka resiko kanker prostatnya High.

Proses fuzzifikasi dari berbagai faktor tersebut dibuat oleh fungsi-fungsi berikut :

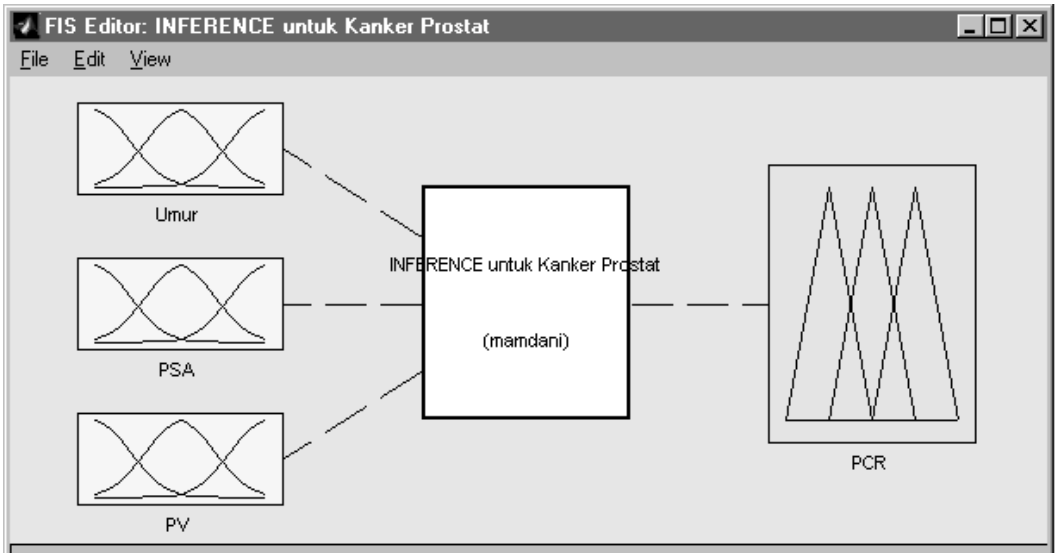
$$PSA(A) = \begin{cases} a, & 0 < a < 16 \\ 1, & a \geq 50 \end{cases} \qquad Umur(B) = \begin{cases} 1, & b \geq 65 \\ b, & 0 < b < 65 \end{cases}$$

$$PV(C) = \begin{cases} c, & 3.8 \leq c \leq 308 \\ 0, & c < 3.8 \\ 1, & c > 308 \end{cases} \qquad PCR(D) = \begin{cases} z, & 0 \leq c \leq 100 \\ 0, & c < 0 \\ 1, & c > 100 \end{cases} \qquad (1)$$

Gambar II.2 Formula 1

Sumber : (Nurul Hidayat, M. Munawar Yusro;2007;4)

Sedangkan FIS yang dikembangkan memiliki struktur sebagai berikut :

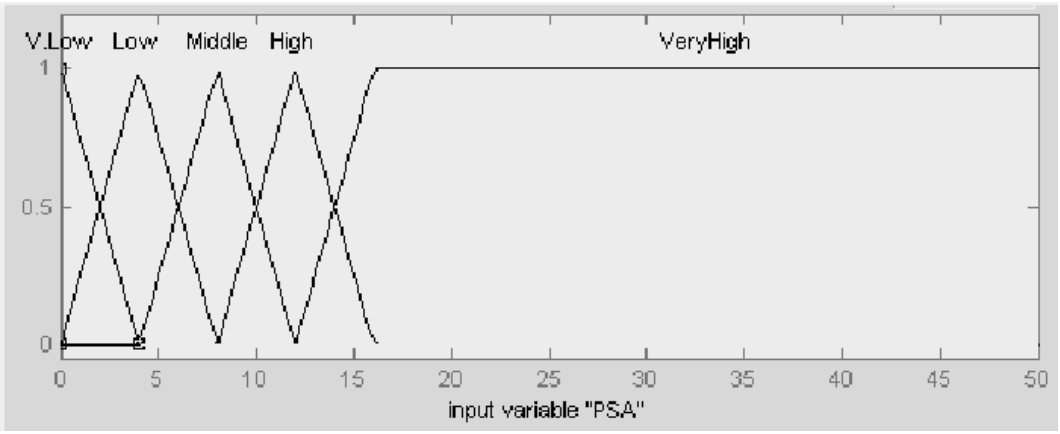


Gambar II.3 Struktur FIS

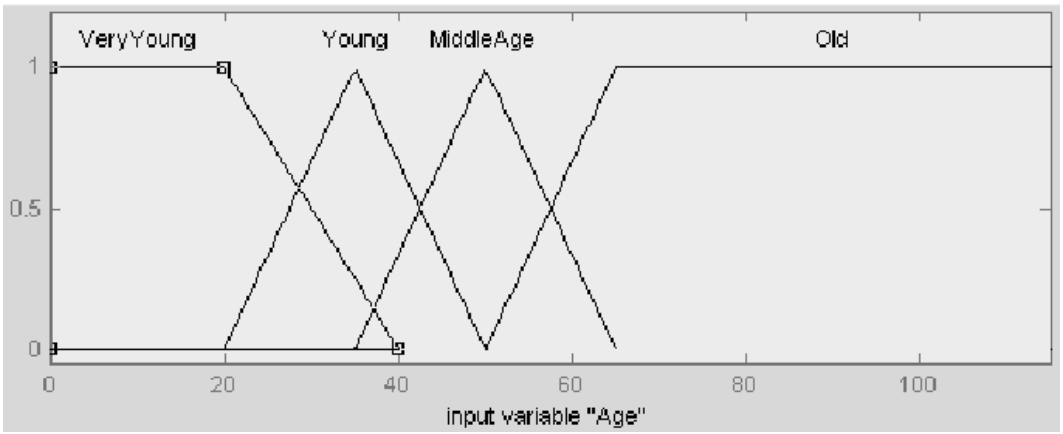
Sumber : (Nurul Hidayat, M. Munawar Yusro;2007;4)

2. Struktur Faktor Fuzzy

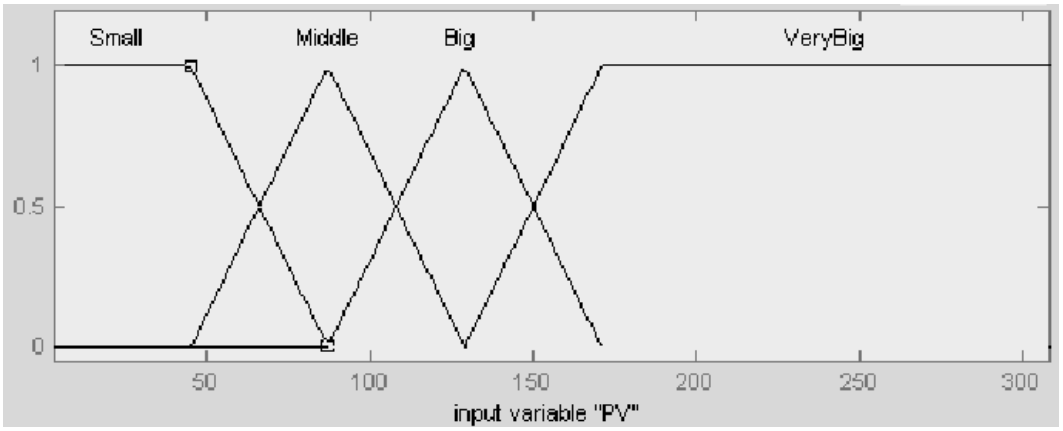
Keanggotaan dari fungsi yang digunakan diperoleh dengan formula (1) dan dilihat dalam gambar berikut.



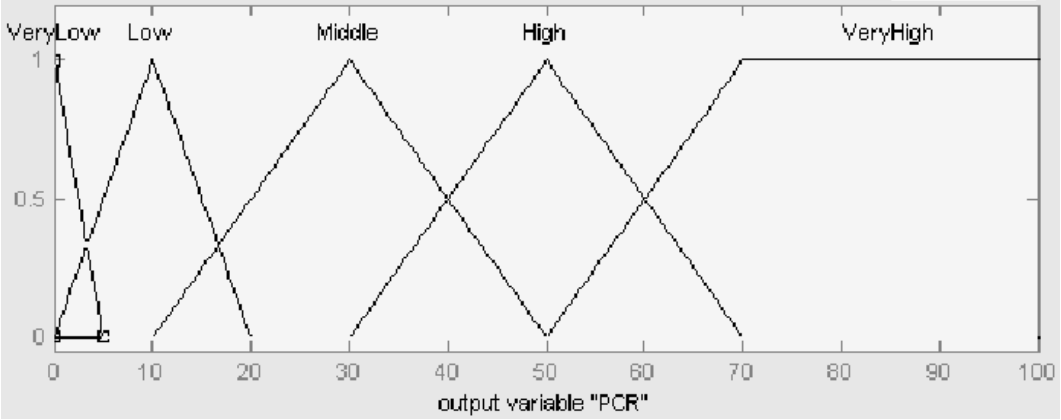
Gambar II.4 Fungsi Keanggotaan PSA



Gambar II.5 Fungsi Keanggotaan Umur Pasien



Gambar II.6 Fungsi Keanggotaan PV



Gambar II.7 Fungsi Keanggotaan PCR

Sumber : (Nurul Hidayat, M. Munawar Yusro;2007;5-6)

Berdasarkan aturan-aturan yang dikembangkan dan formula (1), diperoleh ekspresi linguistik PSA adalah sebagai berikut :

$$\mu_{veryLow} A) = \begin{cases} \frac{4-a}{4}, & 0 < a < 4 \\ 0 \quad , \end{cases} \qquad \mu_{Low}(A) = \begin{cases} \frac{a}{4}, & 0 < a \le 4 \\ \frac{8-a}{4}, & 4 < a < 8 \\ 0 \quad , & a \ge 8 \end{cases}$$

$$\mu_{Middle}(A) = \begin{cases} 0 \, , & a \le 4 \\ \frac{a-4}{4}, & 4 < a \le 8 \\ \frac{12-a}{4}, & 8 < a < 12 \\ 0 \quad , & a \ge 12 \end{cases} \qquad \mu_{High}(A) = \begin{cases} 0, & a \le 8 \\ \frac{a-8}{4}, & 8 < a \le 12 \\ \frac{16-a}{4}, & 12 < a < 16 \\ 0 \quad , & a \ge 16 \end{cases}$$

$$\mu_{VeryHigh}(A) = \begin{cases} 0, & a \le 12 \\ \frac{a-12}{4}, & 12 < a \le 16 \\ 0 \quad , & a \ge 16 \end{cases} \qquad (2)$$

Gambar II.8 Formula 2

Sumber : (Nurul Hidayat, M. Munawar Yusro;2007;6)

Ekspresi linguistik yang lainnya, untuk Umur (very young, Young, Middle Age, and Older) dan PV diperoleh dengan cara yang sama. Sedangkan untuk faktor output PCR, ekspresi linguistiknya adalah Very Low, Low, Middle, High, dan Very High, yang dapat diekpresikan sebagai very Small, Small, Middle, High, dan Very High seperti dalam formula (2). Sebagai contoh, fungsi keanggotaan untuk High PSA, Middle Age, dan Very Big PV memiliki bentuk respektif :

$$\mu_{high}(PSA) = \{0/8 + 0.25/9 + 0.5/10 + 0.75/11 + 1/12 + 0.75/13 + 0.5/14 + 0.25/15 + 0/16\}$$

$$\mu_{Middle}(Umur) = \{0/35 + 0.33/40 + 0.67/45 + 1/50 + 0.67/55 + 0.33/60 + 0/65\}$$

$$\mu_{VeryHigh}(PV) = \{0/2129 + 0.1/133 + 0.2/137 + ... + 1/171 + 1/175 + ... + 1/308\}$$

Gambar II.9 Bentuk Respektif

Sumber : (Nurul Hidayat, M. Munawar Yusro;2007;7)

3. Defuzzifikasi (*Deffuzification*)

Pada tahap ini, derajat kebenaran (*truth degrees* / α) dari aturan ditentukan terhadap tiap-tiap aturan dengan menggabungkan fungsi min dilanjutkan fungsi max antar aturan yang aktif. Sebagai contoh, untuk PSA = 40 ng/ml, umur = 55 thn, PV = 230 ml, aturan 60 dan 80 akan dijalankan sehingga diperoleh :

α_{60} = min(Very High PSA, Middle Umur, Very Big PV) = min(1, 0.67, 1) = 0.67

α_{80} = min(Very High PSA, Old Umur, Very Big PV) = min(1, 0.33, 1) = 0.33

Dengan inferensi max-min Mamdani akan diperoleh fungsi keanggotaan untuk sistem ini yaitu max(60 α , 80 α) = 0.67 , yang berarti very high PCR.

Nilai Crisp PCR dapat dihitung dengan defuzzifier metode *center of gravity* yaitu :

$$D^{\cdot} = \frac{\int D \cdot \mu_{middle}(D)\, dD}{\int \mu_{middle}(D)\, dD} \quad (3)$$

Gambar II.10 Formula 3

Sumber : (Nurul Hidayat, M. Munawar Yusro;2007;7)

Setelah dilakukan perhitungan nilai PCR =78.4, yang mempunyai arti pasien mengidap kanker prostat memiliki kemungkian 78.4 %.

II.4. Visual Basic Net 2010

Visual basic merupakan salah satu bahasa pemrograman yang handal dalam lingkungan Windows. Visual basic telah merajai pasar pembuatan perangkat lunak/*software* sampai beberapa decade tanpa ada yang menyainginya. Visual basic 2010 merupakan teknologi terbaru yang masuk ke dalam visual studio bersama dengan C#, C++, dan lainnya.

Visual basic 2010 dapat digunakan untuk mengembangkan berbagai *software* dengan lingkungan Windows. Namun kini, juga tersedia pada lingkungan Unix (Linux). Banyak kelebihan yang ditawarkan bahasa Visual Basic, diantaranya mempunyai bahasa yang mudah dimengerti, didukung lingkungan pengembangan (IDE) yang canggih, pemrograman berorientasi objek secara penuh, dan lain sebagainya. (Wahana Komputer;2010;iii)

II.5. SQL Server 2008 R2

Microsoft SQL Server 2008 R2 adalah yang paling maju, terpercaya, dan scalable Data platform yang dirilis hingga saat ini. Bangunan pada keberhasilan asli SQL Server 2008 rilis, SQL Server 2008 R2 telah membuat dampak pada organisasi di seluruh dunia dengan perusahaan terobosan kemampuan, memberdayakan pengguna akhir melalui self-service business intelligence (BI), memperkuat efisiensi dan kolaborasi antara database administrator (DBA) dan aplikasi pengembang, dan skala untuk mengakomodasi beban kerja data yang paling menuntut (Ross Mistry,dkk;2010;3).

II.6. UML (*Unified Modeling Language*)

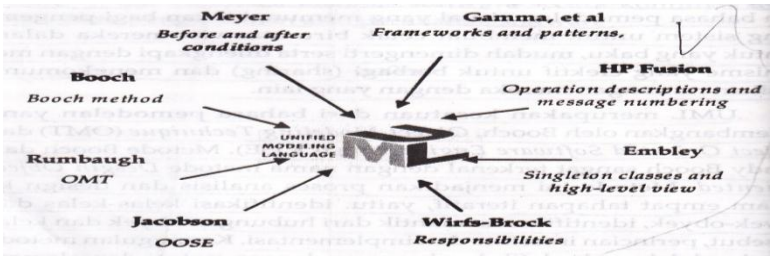
UML(*Unified Modelling Language*) adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi objek. Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembangan sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (Sharing) dan mengkomunikasikan rancangan dengan baik (Munawar;2005:17).

UML merupakan kesatuan bahasa pemodelan yang dikembangkan oleh Booch, *Object Modeling Technique* (OMT) dan *object Oriented Engineering* (OOSE). Metode Booch dari Grady Booch sangat terkenal dengan nama metode *Design Object Oriented.* Metode ini menjadikan proses analisis dan design ke dalam empat tahapan iteratif, yaitu: identifikasi kelas-kelas dan objek-objek,

identifikasi semantik dari hubungan objek dan kelas tersebut, perincian interface dan implementasi. Keunggulan metode Booch adalah pada detil dan kayanya dengan notasi dan elemen.

Pemodelan OMT yang dikembangkan oleh Rumbaugh didasarkan pada analisis terstruktur dan pemodelan *entity-relationship*. Tahapan utama dalam metodologi in adalah analisis, desain sistem, desain objek dan impelmentasi. Keunggulan motode ini adalah dalam penotasian yang mendukung semua konsep OO. Metode OOSE dari Jacobson lebih memberi penekanan dan *use case.* OOSE memiliki tiga tahapan yaitu membuat model *requirement* dan analisis, desain dan implementasi dan model pengujian (test Model). Keunggulan metode ini adalah mudah dipelajari karena memiliki notasi yang sederhananamun mencangkup seluruh tahapan dalam rekayasa perangkat lunak.

Dengan UML, metode Booch, OMT dan OOSE digabungkan dengan membuang elemen-elemen yang tidak praktis ditambah dengan elemen-elemen dari metode lain yang lebih efektif dan elemen-elemen baruyang belum ada pada metode terdahulu sehingga UML lebih ekspresif dan seragam daripada metode lainnya. Unsur-unsur yang membentuk UML ditunjukkan dalam Gambar II.11

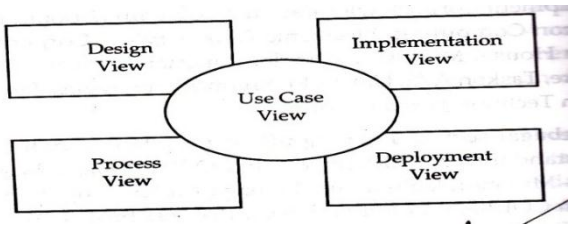


Gambar II.11 Unsur-unsur yang membentuk UML

Sumber: (Munawar;2005:18)

UML adalah hasil kerja dari konsorsium berbagai organisasi yang berhasil dijadikan sebagai standar baku dalam OOAD (*Object Oriented Analysis dan Design*). UML tidak hanya dominan dalam penotasian di lingkungan OO tetapi juga populer di luar lingkungan OO. Ada tiga karakter penting yang melekat di UML yaitu sketsa, cetak biru dan bahasa pemrograman. Sebagai sebuah sketsa UML bisa berfungsi sebagai sebuah cetak biru kerena sangat lengkap dan detil. Dengan cetak biru ini maka akan bisa diketahui informasi detil tentang coding program (*Forward engineering*) atau bahkan membaca program dan mengiterpretasikannya kembali ke dalam diagram (*reverse engineering*). *Reverse engineering* sangat berguna pada situasi dimana kode program yang tidak terdokumentasi asli hilang atau bahkan belum dibuat sama sekali.

Sebagai bahasa pemrogaraman, UML dapat menterjemahkan diagaram yang ada di UML menjadi kode program siap untuk dijalankan. UML dibangun atas model 4+1 *view*. Model ini didasarkan pada fakta bahwa struktur sebuah sistem dideskripsikan dalam *view* dimana salah satu diantaranya *use case view*. *Use case view* ini memegang peran khusus untuk mengintegrasikan *content* ke *view* yang lain. Model 4+1 *view* ditunjukkan pada gambar II.12



Gambar II.12 Model 4+1 *View*

Sumber: (Munawar;2005:20)

Kelima *view* tersebut tidak berhubungan dengan diagram yang dideskripsikan di UML. Setiap *view* berhubungan dengan perspektif tertentu dimana sistem akan diuji. *View* yang berbeda akan menekankan pada aspek yang berbeda dari sistem yang mewakili tentang sistem bisa dibentuk dengan menggabungkan informasi-informasi yang ada pada kelima *view* tersebut.

Use case view mendefinisikan perilaku eksternal sistem. Hal ini menjadi daya tari bagi *end user,* analis dan tester. Pandangan ini mendefinisikan kebutuhan sistem karena mengandung semua *view* yang lain yang mendeskripsikan aspek-aspek tertentu dari peran dan sering dikatakan yang mendrive proses pengembangan perangkat lunak.

Design view mendeskripsikan struktur logika yang mendukung fungsi-fungsi yang dibutuhkan di *use case*. *Design view* ini berisi definisi komponen program, class-class utama bersama-sama dengan spesifikasi data, perilkau dan interaksinya. Informasi yang terkandung di *view* ini menjadi perjatian para programer karena menjelaskan secara detil bagaimana fungsionalitas sistem akan diimplementasikan.

Implemantasi *view* menjelaskan komponen-komponen fisi dari sistem yang akan dibangun. Hal ini berbeda dengan komponen logic yang dideskripsikan pada *design view.* Termasuk disini diantaranya *file exe*, *library* dan *database.* Informasi yang ada di *view* dan integrasi sistem.

Proses *view* berhubungan dengan hal-hal yang berkaitan dengan *concurrency* do dalam sistem. Sedangkan *deployment view* menjelaskan bagaimana komponen-komponen fisik didistribusikan ke lingkungna fisik seperti

jaringan komputer dimana sistem akan dijalankan. Kedua *view* ini menunjukkan kebutuhan non fungsional dari sistem seperti toleransi kesalahan dan hal-hal yang berhubungan dengan kinerja (Munawar;2005:17-21).

1. Use Case Diagram

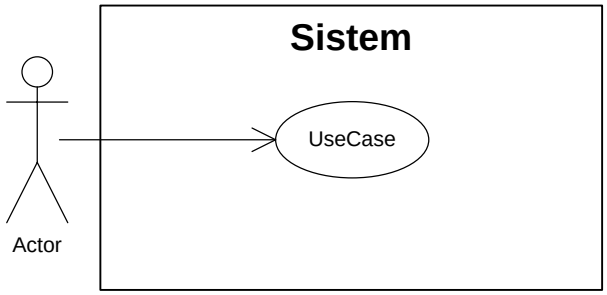
Use case adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara deskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem disebut *scenario.* Setiap *scenario* mendeskripsikan urutan kejadian. Setiap urutan diinisialisasi oleh orang, sistem yang lain, perangkat keras atau urutan waktu. Dengan demikian secara singkat bisa dikatakan *use case* adalah serangkaian *scenario* yang digabungkan bersama-sama oleh tujuan umum pengguna.

Dalam pembicaraan tentang *use case,* pengguna biasanya disebut dengan *actor. Actor* adalah sebuah peran yang bisa dimainkan oleh pengguna dalam interaksinya dengan sistem.

Model *use case* adalah bagai dari model *requirement.* Termasuk disini adalah problem domain object dan penjelasan tentang *user interface*. *Use case* memberikan spesifikasi fungsi-fungsi yang ditawarkan oleh sistem dari *perspectif user.*

Notasi *use case* menunjukkan 3 aspek dari sistem yaitu *actor use case* dan *system/sub system boundary. Actor* mewakili peran orang, *system* yang lain atau

alat ketika berkomunikasi dengan *use case.* Ilustrasi *actor*, *usecase* dan *system* ditunjukkan pada gambar II.13



Gambar II.13 *Usecase* Diagram

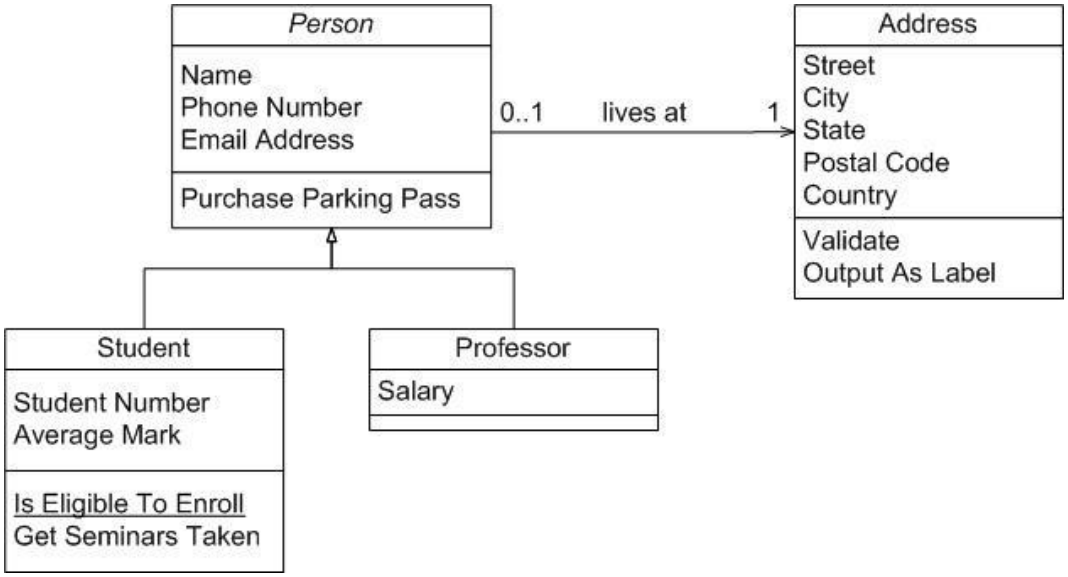
Sumber: (Munawar;2005:64)

Untuk mengidentifikasi *actor,* harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks targer sistem. *Actor* adalah *abstraction* dari orang dan sistem yang lain yang mengaktifkan fungsi dari target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa *actor* berinteraksi dengan *use case*, tetapi tidak memiliki kontrol atas *use case.*

Use case adalah abstraksi dari interaksi antara sistem dan *actor.* Oleh karena itu sangat penting untuk memilih abstraksi yang cocok. *Use case* dibuat berdasarkan keperluan *actor. Use case* harus merupakan ‘apa’ yang dikerjakan *software* aplikasi, bukan ‘bagaimana’ *software* aplikasi mengerjakannya. Setiap *use case* harus diberi nama yang menyatakan apa hal yang dicapai dari hasil interaaksinya dengan *actor*. Namun *use case* boleh terdiri dari beberapa kata dan tidak boleh ada dua *use case* yang memiliki nama yang sama (Munawar;2005:63-66).

2. Class Diagram

Class diagram sangat membantu dalam visualisasi struktur kelas dari suatu sistem. Hal ini disebabkan karena *class* adalah deskripsi kelompok obyek-obyek dengan *property*, perilaku (operasi) dan relasi yang sama. Disamping itu *class diagram* memberikan pandangan global atas sebuah sistem. Hal tersebut tercermin dari *class-class* yang ada dan relasinya satu dengan yang lainnya. (Munawar; 2011;219) Contoh diagram *class* dapat dilihat pada gambar II.14 dibawah ini:



Gambar II.14 *Class* Diagram

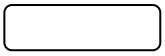
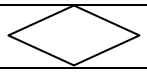
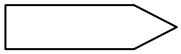
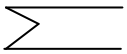
Sumber: (Munawar;2005:220)

3. Activity Diagram

Activity Diagram adalah teknik untuk mendiskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity Diagram* mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaannya dengan *flowchart* adalah *activity* diagram bisa mendukung perilaku paralel sedangkan flowchart tidak

bisa.(Munawar;2005;109) Adapun simbol *activity diagram* dapat dilihat pada table II.3 :

Tabel II.3 Simbol *Activity Diagram*

Notasi	Keterangan
	Titik Awal
	Titik Akhir
	*Activity*
	Pilihan untuk pengambilan keputusan
	*Fork* digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu
	*Rake* menunjukkan adanya dekomposisi
	Tanda waktu
	Tanda pengiriman
	Tanda penerimaan
	Aliran Akhir (*Flow Final*)

Sumber : (Munawar;2005;109-110)

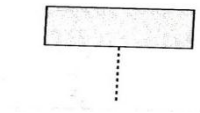
4. Squence Diagram

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah sekenario. Diagram ini menunjukkan sejumlah contoh objek dan pesan yang diletakkan diantaran objek-objek ini di dalam *use case.*

Komponen utama *sequence diagram* terdiri atas objek yang ditulisakan dengan kotak segiempat bernama. *Messege* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan *progress vertical.*

a. Objek /*participant*

Objek diletakkan di dekat bagian atas diagram dengan urutan dari kiri ke kanan. Mereka diatur dalam urutan guna menyederhanakan diagram. Setiap *participant* dihubungkan dengan garis titik-titik yang disebut *lifeline.* Sepanjang *lifeline* ada kotak yang disebut *activation. Activation* mewakili sebuah eksekusi operasi dari *participant.* Panjang kotak ini berbanding lurus dengan durasi *activation.* Bentuk *participant* dapat dilihat pada gambar II.15



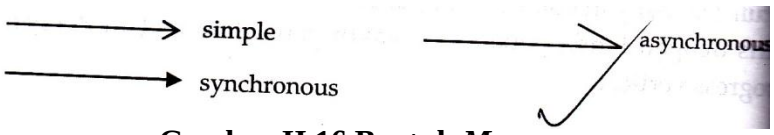
Gambar II.15 Bentuk *Participant*

Sumber: (Munawar;2005:88)

b. *Messege*

Sebuah *messsage* bergerak dari satu *participant* ke *participant* yang lain dan dari satu *lifeline* ke *lifeline* yang lain. Sebuah *participant* bisa mengirim sebuah *message* kepada dirinya sendiri.

Sebuah *message* bisa jadi *simple, synchronous* atau *asynchoronous. Message* yang *simple* adalah sebuah perpindahan (transfer), contoh dari satu *participant* ke *participant* yang lainnya. Jika sebuah *participant* mengirimkan sebuah *messagae* tersebut akan ditunggu sebelum diproses dengan urusannya. Namun jika *message asynchoronous* yang dikirimkan, maka jawabannya atas *message* tersebut tidak perlu ditunggu. Simbol *message* pada *squnence diagram* dapat dilihat pada gambar II.16



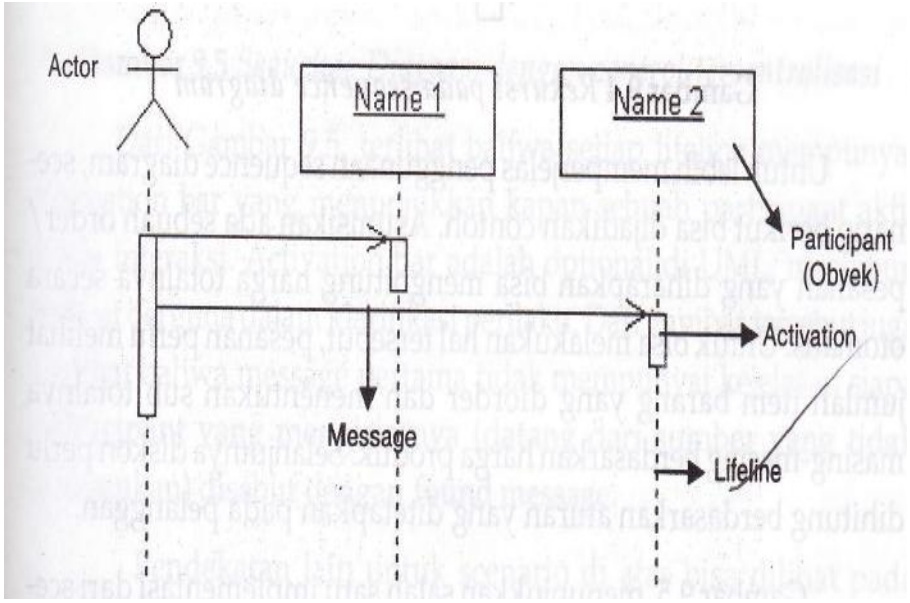
Gambar II.16 Bentuk *Messege*

Sumber: (Munawar;2005;88)

c. Time

Time adalah diagram yang mewakili waktu pada arah vertikal. Waktu dimulai dari ata ke bawah. *Message* yang lebih dekat dari atas akan dijalankan terlebih dahulu dibanding *message* yang lebih dekat ke bawah.

Terdapat dua dimensi pada *squence diagram* yaitu dimensi dari kiri ke kanan menunjukkan tata letak *participant* dan dimensi dari atas ke bawah menunjukkan lintasan waktu (Munawar;2005;87-89). Simbol-simbol yang ada pada *squence diagram* ditunjukkan pada gambar II.17



Gambar II.17 *Sequence Diagaram*

Sumber: (Munawar,2005:89)

II.7. Model Data Entity Relationship

Entity Relationship Model(ERM) merupakan suatu model data yang dikembangkan berdasarkan obyek. ERM digunakan untuk menjelaskan hubungan antar data dalam basis data kepada pengguna secara logik. ERM didasarkan pada suatu persepsi bahwa *real world* terdiri atas obyek-obyek dasar yang mempunyai hubungan/kerealisan antar obyek-obyek dasar tersebut. ERM digambarkan dalam bentuk diagram yang disebut diagram ER (ER diagram/ERD).

1. Entitas (*Entity*)

Entitas merupakan obyek-obyek dasar yang terkait di dalam sistem. Obyek dasar dapat berupa orang, benda atau hal yang keterangannya perlu disipan di dalam basis data. Untuk menggambarkan entitas digunakan aturan sebagai berikut:

a. Entitas dinyatakan dalam simbol persegi panjang,
b. Nama entitas dituliskan di dalam simbol persegi panjang,
c. Nama entitas berupa kata benda, tunggal, dan
d. Nama entitas sedapat mugnkin mengguanakan nama yang mudah dipahami dan dapat menyatakan maknanya dengan jelas. (Edhy Sutanta;2011;91-92)

Tabel II.4 Contoh Entitas

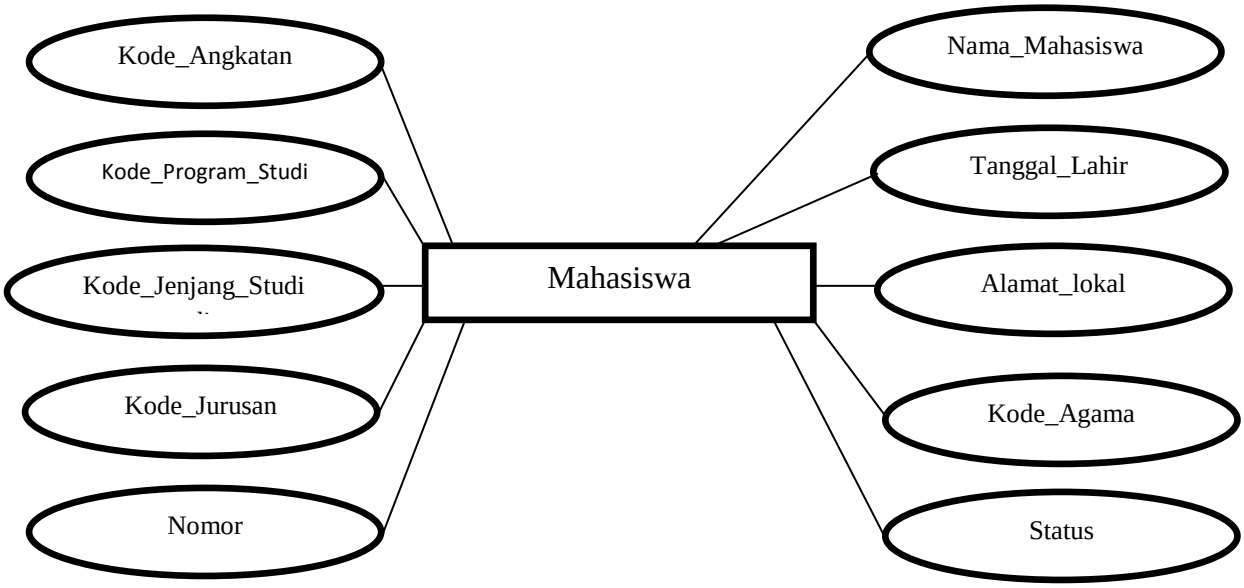
Objek Dasar	Simbol Entitas
Mahasiswa	Mahasiswa
Dosen	Dosen
Wali Mahasiswa/Orang Tua	Wali Mahasiswa

Sumber : (Edhy Sutanta;2011;94)

2. Atribut (*Attribute*)

Atribut sering pula disebut sebagai property (*property*), merupakan keterangan-keterangan yang terkait pada sebuah entitas yang perlu disimpan dalam bsis data. Atribut berfungsi sebagai penjelas pada sebuah entitas. Untuk menggabarkan atribut digunakan aturan sebagai berikut :

a. Atribut dinyatakan dengan simbol elips
b. Nama atribut dituliskan di dalam simbol elips
c. Nama atribut berupa kata benda, tunggal
d. Nama atribut sedapat mungkin menggunakan nama yang mudah dipahami dan dapat menyatakan maknanya dengan jelas.
e. Atribut dihubungkan dengan entitas yang bersesuaian yang bersesuaian dengan menggunakan sebuah garis. (Edhy Sutanta;2011;99)

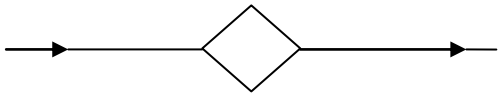
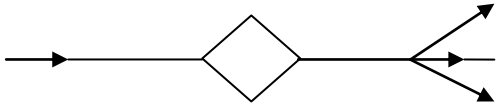
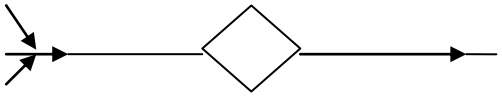
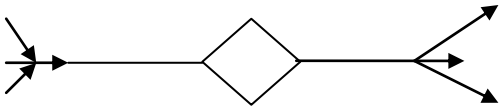


Gambar II.18 Contoh Atribut pada entitas Mahasiswa

Sumber : (Edhy Sutanta;2011;100)

3. Kerelasian Antar Entitas (*Relationship*)

Kerelasian antar entitas mendefinisikan hubungan antara dua buah entitas. Kerelasian adalah kejadian atau transaksi yang terjadi di antara dua buah entitas yang keterangannya perlu disimpan dalam basis data. (Edhy Sutanta;2011;101).

Jenis Kerelasian	Simbol yang digunakan
1 ke 1 :	
1 ke -n	
-n ke 1	
-n ke –n	

Gambar II.19 Simbol kerelasian antar entitas

Sumber : (Edhy Sutanta;2011;110)

4. Normalisasi

Perancangan basis data menghasilkan sekumpulan relasi yang salig berkerelasian dalam lingkup sebuah sistem. Untuk memenuhi batasan dalam definisi basis data maka setiap rancangan relasi perlu diuji untuk menentukan apakah relasi tersebut telah optimal. Pengujian dilaukukan berdasarkan criteria tertentu. Jika relasi belum optimal maka perlu dilakukan proses normalisasi adalah dekomposisi relasi menjadi relasi-relasi baru yang sederhana.(Edhy Sutanta;2011;174)

a. Relasi bentuk normal pertama(1NF)

   Relasi yang disebut sebagai 1NF jika memenuhi criteria sebagai berikut :

   1. Jika seluruh atribut dalam relasi bernilai atomik (*atomic value*)
   2. Jika seluruh atribut dalam relasi bernilai tunggal (*single value*)
   3. Jika relasi tidak memuat set atribut berulang
   4. Jika semua record mempunyai sejumlah atribut yang sama

b. Relasi bentuk normal kedua(2NF)

   Relasi disebut 2NF jika memenuhi kriteria sebagai berikut :

   1. Jika memenuhi criteria 1NF
   2. Jika semua atribut nonkunci FD pada PK

c. Relasi bentuk normal ketiga(3NF)

   Suatu relasi disebut sebagai 3NF jika memenuhi criteria sebagai berikut :

   1. Jika memenuhi criteria 2NF
   2. Jika setiap atribut nonkunci tidak TDF(*non transitive dependency*).(Edhy Sutanta;2010;176-177)