BAB II

LANDASAN TEORI

II.1. Sistem Pakar

Menurut Sutabri (2010:14), sistem dapat juga diartikan sebagai kumpulan dari entitas seperti manusia, saran, menentukan proses secara teratur, saling mempengaruhi atau saling bersaing satu dengan yang lainnya, dimana keseluruhannya merupakan satu kesatuan untuk mencapai tujuan yang telah ditetapkan bersama. Menurut Kusrini (2009:2), pakar yang dimaksud di sini adalah orang yang mempunyai keahlian khusus yang dapat menyelesaikan masalah yang tidak dapat diselesaikan oleh orang awam.

Dalam buku Perancangan Sistem Pakar (2012:1), Nita dan Rahmat mengungkapkan beberapa definisi sistem pakar menurut beberapa ahli yaitu sebagai berikut :

1. Menurut Durkin : Sistem pakar adalah suatu program komputer yang dirancang untuk memodelkan kemampuan penyelesaian masalah yang dilakukan seorang pakar.
2. Menurut Ignizo : Sistem pakar adalah suatu model dan prosedur yang berkaitan dalam suatu domain tertentu , yang mana tingkat keahliannya dapat dibandingkan dengan keahlian seorang pakar.
3. Menurut Giarratano dan Riley : Sistem pakar adalah suatu sistem komputer yang bisa menyamai atau meniru kemampuaan seorang pakar.

Sistem pakar sebagai sebuah program yang difungsikan untuk menirukan pakar manusia harus bisa melakukan hal-hal yang dapat dikerjakan oleh seorang pakar (Kusrini,2009:3). Pengetahuan merupakan sumber utama yang sangat penting, tetapi hanya dimiliki oleh sedikit pakar saja. Oleh karena itu penting untuk memperoleh kepakaran itu agar setiap orang bisa menggunakannya. Sistem pakar merupakan media langsung untuk melakukan pekerjaan seorang pakar.

Karakteristik yang paling umum dalam suatu sistem pakar adalah sistem ini menggunakan basis pengetahuan yang besar. Karena dalam sistem pakar ini peranan pengetahuan sangat penting, maka sering juga disebut sebagai sistem berbasis pengetahuan (*knowledge based system*). Proses pembuatan sistem pakar disebut sebagai rekayasa pengetahuan (*knowledge engineering*) serta orang yang membuat sistem pakar disebut sebagai *knowledge engineer* (Andi, 2009:3).

II.1.1. Definisi dan Tujuan Sistem Pakar

Sistem pakar adalah suatu sistem yang berusaha mengadopsi pengetahuan manusia ke komputer, agar komputer dapat menyelesaikan masalah seperti yang dilakukan oleh para ahli. Sistem pakar yang baik dirancang agar dapat menyelesaikan suatu permasalahan tertentu dengan meniru kerja dan membandingkan tingkat keahliannya dari para ahli (Sri Hartati dan Sari Iswanti, 2009:2).

Sistem pakar adalah sebuah teknik inovatif baru dalam menangkap dan memadukan pengetahuan. Kekuatan sistem ini terletak dalam kemampuannya memecahkan masalah-masalah praktis saat para ahli atau pakar berhalangan. Kemampuan yang dimiliki sistem pakar disebabkan terdapatnya basis

pengetahuan yang berupa pengetahuan non-formal yang sebagian besar berasal dari pengalaman.

Tujuan pengembangan sistem pakar sebenarnya bukan untuk menggantikan peran manusia, tetapi untuk mensubstitusikan pengetahuan manusia ke dalam bentuk sistem, sehingga dapat digunakan oleh orang banyak.

Ada banyak manfaat yang dapat diperoleh dengan memanfaatkan keahlian didalam sistem pakar, antara lain :

1. Masyarakat awam non-pakar dapat memanfaatkan keahlian didalam bidang tertentu tanpa kehadiran langsung seorang pakar.
2. Meningkatkan produktivitas kerja, yaitu bertambah efisiensi pekerjaan tertentu serta hasil solusi kerja.
3. Penghematan waktu dalam menyelesaikan masalah yang kompleks.
4. Memberikan penyederhanaan solusi untuk kasus-kasus yang kompleks dan berulang-ulang.
5. Pengetahuan dari seorang pakar dapat didokumentasikan tanpa ada batas waktu.
6. Memungkinkan penggabungan berbagai bidang pengetahuan dari berbagai pakar untuk dikombinasikan.

II.1.2. Ciri dan Karateristik Sistem Pakar

Ada beberapa ciri dan karakteristik yang membedakan sistem pakar dengan sistem yang lain. Ciri dan karakteristik ini menjadi pedoman utama dalam pengembangan sistem pakar. Ciri dan karakteristik sistem pakar adalah sebagai berikut :

1. Pengetahuan sistem pakar merupakan suatu konsep, bukan berbentuk numerik. Hal ini dikarenakan komputer melakukan proses pengolahan data secara numerik sedangkan keahlian dari seorang pakar adalah fakta dan aturan-aturan, bukan numerik.
2. Informasi dalam sistem pakar tidak selalu lengkap, subyektif, tidak konsisten, subyek terus berubah dan tergantung pada kondisi lingkungan sehingga keputusan yang diambil bersifat tidak pasti dan tidak mutlak “ya” atau “tidak” . Akan tetapi, keputusan yang diambil sesuai dengan kebenaran tertentu. Oleh karena ini dibutuhkan kemampuan sistem untuk belajar secara mandiri dalam menyelesaikan masalah-masalah dengan pertimbangan-pertimbangan khusus.
3. Kemungkinan solusi sistem pakar terhadap suatu permasalahan adalah bervariasi dan mempunyai banyak pilihan jawaban yang dapat diterima. Semua faktor yang ditelusuri memiliki ruang masalah yang luas dan tidak pasti. Oleh karena itu diperlukan fleksibilitas sistem dalam menangani kemungkinan solusi dari berbagai permasalahan.
4. Perubahan atau pengembangan pengetahuan dalam sistem pakar dapat terjadi setiap saat bahkan sepanjang waktu sehingga diperlukan kemudahan dalam modifikasi sistem untuk menampung jumlah pengetahuan yang semakin besar dan semakin bervariasi.
5. Pandangan dan pendapat setiap pakar tidaknya selalu sama. Oleh karena itu, tidak ada jaminan bahwa solusi sistem pakar merupakan jawaban yang pasti benar. Setiap pakar akan memberikan pertimbangan-pertimbangan berdasarkan faktor subyektif.

6. Keputusan merupakan bagian terpenting dari sistem pakar. Sistem pakar harus memberikan solusi yang akurat berdasarkan masukan pengetahuan meskipun solusinya sulit sehingga fasilitas informasi sistem harus selalu diperlukan.

Berikut ini merupakan perbandingan antara sistem pakar, sistem konvensional dan kepakaran manusia, seperti pada tabel II.1.

Tabel II.1 Perbandingan antara Sistem Pakar, sistem Konvensional dan kepakaran manusia

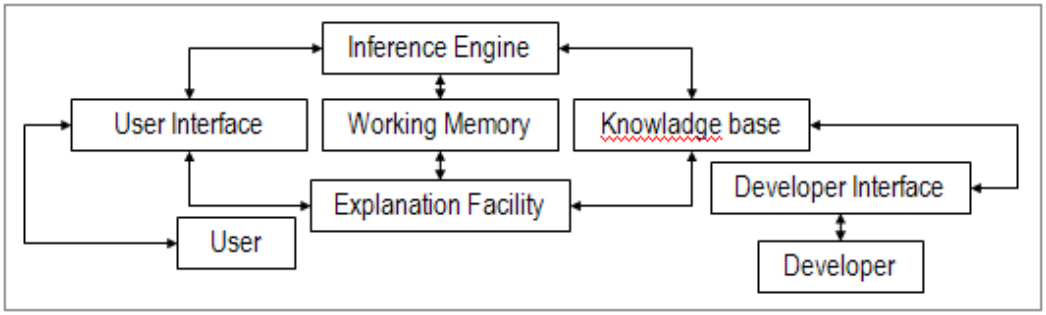
Kepakaran Manusia	**Sistem Pakar**	**Sistem Konvensional**
Menggunakan pengetahuan dalam bentuk *rules* atau secara *heuristics* untuk memecahkan permasalahan	Memproses pengetahuan yang terekspresikan dalam bentuk *rules* dan menggunakan penalaran simbolis untuk memecahkan permasalahan	Memproses data dan menggunakan algoritma untuk memecahkan permasalahan numeris
Pengetahuan terdapat pada otak manusia	Terdapat pemisahan yang jelas antara pengetahuan dan pemrosesan	Tidak memisahkan pengetahuan dari struktur control dalam memproses pengetahuan tersebut
Mampu menjelaskan garis besar dan detil penalaran	Menelusuri *rules* yang terpicu dalam proses Tanya-jawab dan menjelaskan bagaimana konklusi di capai.	Tidak menjelaskan bagaimana konklusi di capai
Menggunakan penalaran dari informasi yang tidak pasti, tidak lengkap dan kabur (*fuzzy*)	Memungkinkan penalaran yang tidak pasti dari data yang tidak lengkap, tidak pasti dan kabur (*fuzzy*)	Menggunakan data yang lengkap dan pasti
Dapat membuat kesalahan jika informasi tidak lengkap atau kabur	Dapat membuat kesalahan jika informasi tidak lengkap atau kabur	Tidak menghasilkan solusi apapun atau menghasilkan solusi yang salah jika data tidak lengkap atau kabur

Melalui proses pembelajaran bertahun untuk meningkatkan kemampuan pemecahan masalah	Kemampuan pemecahan masalah dapat ditingkatkan cukup dengan menambahkan kaidah-kaidah yang baru	Memodifikasi *code program* untuk meningkatkan kemampuan pemecahan masalah

(Sumber : Nita dan Rahmat, 2012: 9)

II.1.3. Arsitektur Sistem Pakar

Arsitektur sistem pakar dapat dilihat pada gambar II.1 dibawah ini



Gambar II.1 Arsitektur Sistem Pakar

(Sumber : Nita dan Rahmat, 2012: 11)

Adapun penjelasan gambar 2.1 adalah sebagai berikut :

1. Mesin inferensi (*Inference Engine*), merupakan otak dari sistem pakar yang mengandung mekanisme fungsi berpikir serta metodologi yang digunakan untuk melakukan penalaran terhadap informasi-informasi dalam basis pengetahuan dan memformulasikan konklusi.
2. Tempat penyimpanan sementara (*Working Memory*), merupakan tempat penyimpanan fakta-fakta yang diketahui dari hasil menjawab pertanyaan.

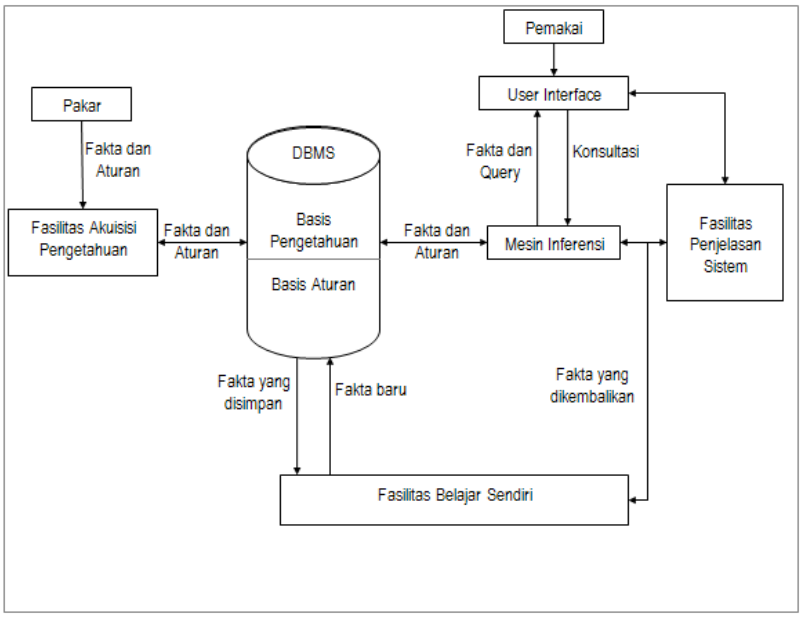
3. Fasilitas penjelasan (*Explanation Facility*), merupakan suatu fasilitas yang memberikan penjelasan saat user mengetahui apakah yang menjadi solusi dari suatu permasalahan.
4. Basis pengetahuan (*Knowladge Base*), merupakan representasi pengetahuan dari seorang atau beberapa pakar yang diperlukan untuk memahami, memformulasikan dan memecahkan masalah, dimana basis pengetahuan terdiri dari dua elemen yaitu fakta dan aturan.
5. Pemakai (*User*), merupakan orang yang menggunakan sistem.
6. Pembangun sistem (*Developer*), merupakan orang yang berhak melakukan perubahan terhadap sistem.

II.1.4. Komponen Sistem Pakar

Suatu sistem disebut sebagai sistem pakar jika mempunyai ciri dan karakteristik tertentu. Hal ini juga harus didukung oleh komponen-komponen sistem pakar yang mampu menggambarkan tentang ciri dan karakteristik tersebut. Komponen-komponen sistem pakar adalah sebagai berikut (Sri Hartati dan Sari Iswanti, 2009:5) :

1. Basis Pengetahuhan (*Knowladge Base*)
2. Mesin Inferensi (*Inference Machine*)
3. Fasilitias Penjelasan (*Explanation Facility*)
4. Fasilitas Akuisisi Pengetahuan (*Knowladge Acquisition Facility*)
5. Fasilitas Pelatihan Sendiri (*Self-Training Facility*)
6. Antar Muka Pemakai (*User Interface*)

Pada gambar II.2 menunjukkan komponen-komponen dari sebuah sistem pakar.



Gambar II.2 Komponen Sebuah Sistem Pakar

(Sumber : Sri Hartati dan Sari Iswanti, 2009: 6)

Adapun penjelasan Gambar II.2 adalah sebagai berikut :

1. Basis pengetahuan (*Knowladge Base*)

Basis pengetahuan merupakan kumpulan pengetahuan bidang tertentu pada tingkatan pakar dalam format tertentu. Pengetahuan ini diperoleh dari akumulasi pengetahuan pakar dan sumber lain yang dapat berupa buku, majalah, jurnal ilmiah dan sebagainya.

2. Mesin Inferensi (*Inference Machine*)

Mesin inferensi adalah bagian dari sistem pakar yang melakukan penalaran dengan menggunakan isi daftar aturan berdasarkan urutan dan pola tertentu. Selama proses konsultasi antar sistem dan pemakai, mesin inferensi menguji aturan satu demi satu sampai kondisi aturan itu benar.

Secara umum ada dua teknik utama yang digunakan dalam mesin inferensi untuk pengujian aturan, yaitu penalaran maju (*forward reasoning*) dan penalaran mundur (*reverse reasoning*).Dalam penalaran maju, aturan-aturan diuji satu demi satu dalam urutan tertentu. Urutan ini mungkin berupa urutan pemasukkan aturan kedalam basis aturan atau juga urutan lain yang ditentukan oleh pemakai. Saat tiap aturan diuji, sistem pakar akan mengevaluasi apakah kondisinya benar atau salah. Jika kondisinya benar, maka aturan itu disimpan kemudian aturan berikutnya diuji. Sebaliknnya kondisinya salah, aturan ini tidak disimpan dan aturan berikutnya diuji.

3. Fasilitas Penjelasan (*Explanation Facility*)

Fasilitas penjelasan sistem merupakan bagian dari sistem pakar yang memberikan penjelasan tentang bagaimana program dijalankan, apa yang harus dijelaskan kepada pemakai tentang suatu masalah, memberikan rekomendasi kepada pemakai, mengakomodasi kesalahan pemakai dan menjelaskan bagaimana suatu masalah terjadi.

Fasilitas penjelasan sistem harus mampu menjelaskan bagaimana harus memeriksa sekering yang putus atau bagaimana memeriksa aki motor, sehingga pemakai dapat mengerti dengan jelas apa yang harus dilakukannya. Dalam sistem pakar, fasilitas penjelasan sistem sebaiknya diintegrasikan ke dalam tabel basis pengetahuan dan basis aturan karena hal ini lebih memudahkan perancangan sistem.

4. Fasilitas Akuisisi Pengetahuan (*Knowladge Acquisition Facility*)

Fasilitas ini merupakan suatu proses untuk mengumpulkan data-data pengetahuan akan suatu masalah dari pakar. Bahan pengetahuan dapat ditempuh

dengan beberapa cara, misalnya mendapatkan pengetahuan dari buku, jurnal ilmiah, para pakar dibidangnya, laporan, literature dan seterusnya. Sumber pengetahuan tersebut dijadikan dokumentasi untuk dipelajari, diolah dan diorganisasikan secara terstruktur menjadi basis pengetahuan.

Sumber pengetahuan tersebut harus dapat diperoleh dengan kemampuan untuk mengolah data-data tersebut menjadi solusi yang efisien, komunikasi yang baik dan kerja sama tim yang solid. Karena itu semua kemampuan menjadi hal yang mutlak diperlukan oleh seorang pengembang sistem.

5. Fasilitas Belajar Sendiri (*Self-training Facility*)

Pakar manusia selalu berusaha untuk memutahirkan pengetahuannya. Ketika sistem pakar menemukan data baru selama proses inferensi, informasi ini dapat ditambahkan dalam basis pengetahuan. Menggunakan fasilitas pelatihan, sistem pakar menerima fakta baru dan membandingkannya dengan fakta yang telah ada. Jika fakta baru yang ditemukan tersebut tidak terdapat dalam domain basis pengetahuan, maka fakta baru tersebut merupakan kandidat yang dapat dimasukkan dalam basis pengetahuan.

6. Antar Muka Pemakai (*User Interface*)

Antar muka pemakai memberikan fasilitas komunikasi antara pemakai dan sistem, memberikan berbagai fasilitas informasi dan berbagai keterangan yang bertujuan untuk membantu mengarahkan alur penelusuran masalah sampai ditemukan solusi.

II.1.5. Bidang-Bidang Pengembangan Sistem Pakar

Ada beberapa bidang pengembangan sistem pakar yang ditunjukkan pada tabel II.2

Tabel II.2 Bidang-bidang Pengembangan Sistem Pakar

Bidang	Penjelasan
Konfigurasi	Merakit komponen yang benar pada sebuah sistem dengan cara yang- benar
Diagnosis	Penelusuran permasalahan berdasarkan bukti-bukti hasil observasi
Instruksi	Memberikan instruksi dan pengajaran tertentu terhadap suatu topic-permasalahan
Interpretasi	Menjelaskan data hasil observasi
Monitor	Membandingkan data observasi dengan data yang diharapkan dan-menganalisanya
Perencanaan	Merencanakan suatu pekerjaan

(Sumber : Nita dan Rahmat, 2012:13)

II.1.6. Keuntungan dan Kelemahan Sistem Pakar

Ada banyak manfaat yang dapat diperoleh dengan mengembangkan sistem pakar, antara lain :

1. Meningkatkan produktivitas kerja, yaitu bertambah efisiensi pekerjaan tertentu serta hasil solusi kerja.
2. Penghematan waktu dalam menyelesaikan masalah yang kompleks.
3. Memberikan penyederhanaan solusi untuk kasus-kasus yang kompleks dan berulang-ulang.
4. Pengetahuan dari seorang pakar dapat didokumentasikan tanpa ada batas waktu.
5. Memungkinkan penggabungan berbagai bidang pengetahuan dari berbagai pakar untuk dikombinasikan.

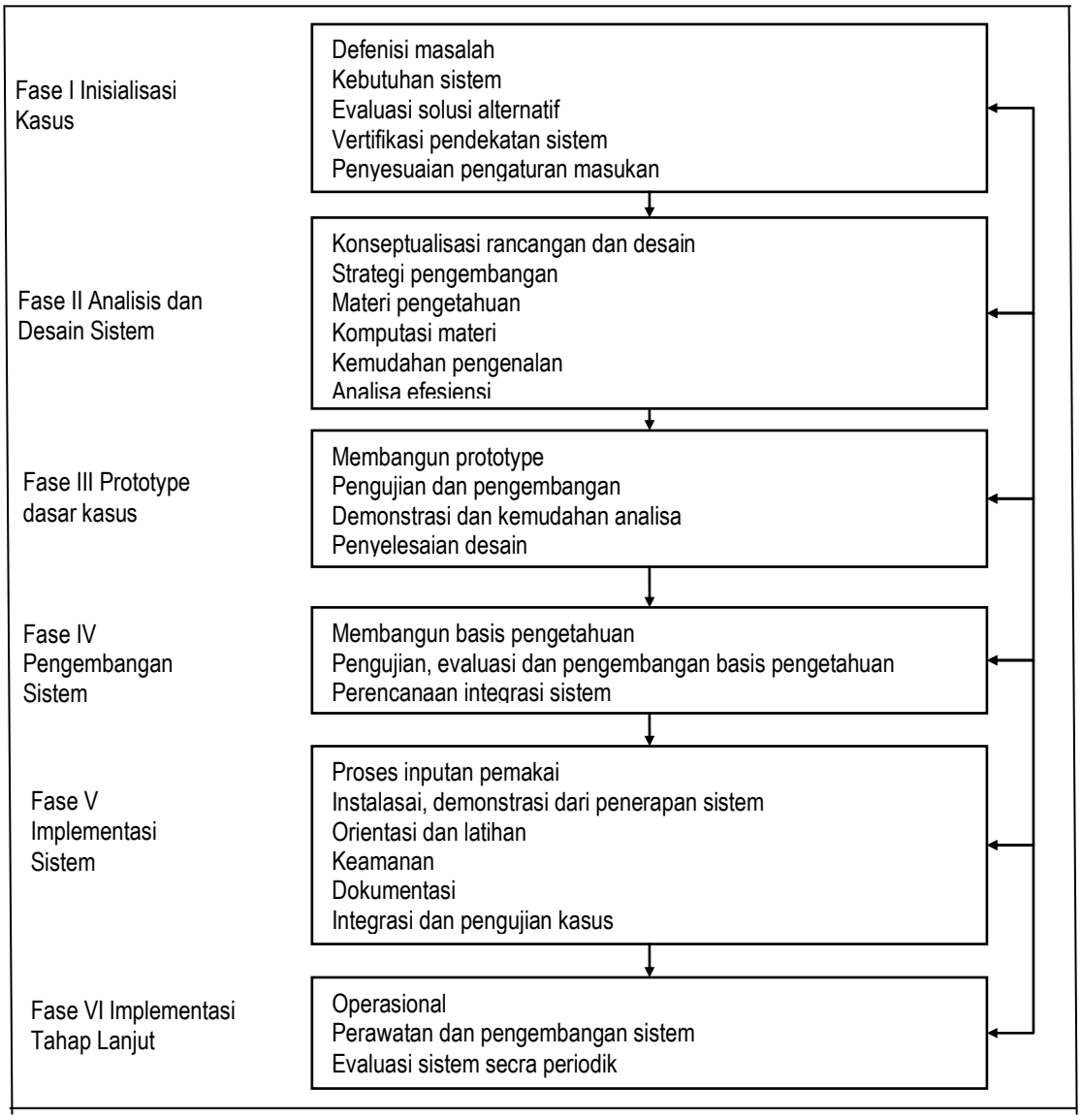
Selain banyak manfaat yang diperoleh, ada juga kelemahan pengembangan sistem pakar, yaitu :

1. Daya kerja dan produktivitas manusia menjadi berkurang karena semuanya dilakukan secara otomatis oleh sistem.
2. Pengembangan perangkat lunak sistem pakar lebih sulit dibandingkan dengan perangkat lunak konvensional.

II.1.7. Tahapan Pengembangan Sistem Pakar

Terdapat 6 fase pengembangan sistem pakar seperti diperlihatkan pada gambar 2.3. Fase-fase tersebut adalah :

1. Fase Inisialisasi Kasus
2. Fase Analisis dan Desain Sistem
3. Fase *Prototype* Dasar Kasus
4. Fase Pengembangan Sistem
5. Fase Implementasi Sistem
6. Fase Implementasi Tahap Lanjut



Gambar II.3 Tahapan Pengembangan Sistem Pakar

(Sumber : Andi,2009:23)

Adapun penjelasan gambar 2.3 sebagai berikut :

1. Identifikasi

Tahap ini merupakan tahap untuk mengkaji dan membatasi masalah yang akan diimplementasikan dalam sistem. Setiap masalah yang diidentifikasi harus di cari solusi, fasilitas yang akan dikembangkan, penentuan jenis bahasa pemrograman dan tujuan yang ingin dicapai dari proses pengembangan tersebut.

2. Konseptualisasi

Hasil identifikasi masalah dikonseptualisasikan dalam bentuk relasi antar data, hubungan antar pengetahuan dan konsep-konsep penting dan ideal yang akan diterapkan dalam sistem. Konseptualisasi juga menganalisis data-data penting yang harus didalami bersama dengan pakar dibidang permasalahan tersebut.

3. Formalisasi

Pada tahap formalisasi konsep-konsep tersebut diimplementasikan secara formal, misalnya memberikan ketegori sistem yang akan dibangun, mempertimbangkan beberapa faktor pengambilan keputusan seperti keahlian manusia, kesulitan dan tingkat kesulitan yang mungkin terjadi, dokumentasi kerja dan sebagainya.

4. Implementasi

Tahap implementasi dimulai dengan membuat garis besar masalah kemudian memecahkan masalah ke dalam modul-modul. Untuk memudahkan maka harus diidentifikasi :

a. Apa saja yang menjadi inputan
b. Bagaimana prosesnya digambarkan dalam bagan alur dan basis aturannya.
c. Apa saja yang menjadi *output* atau hasil dan kesimpulannya.

Sesudah itu semuanya diubah dalam bahasa yang mudah dimengerti oleh komputer dengan menggunakan tahapan fase seperti pada gambar 2.3.

5. Evaluasi

Sistem pakar yang selesai dibangun, perlu untuk dievaluasi untuk menguji-

dan Menemukan kesalahannya. Dalam evaluasi akan ditemukan bagian-bagian yang harus dikoreksi untuk menyamakan permasalahan dan tujuan akhir pembuatan sistem.

6. Pengembangan sistem

Pengembangan sistem bertujuan agar sistem yang dibangun tidak menjadi usang dan investasi sistem tidak sia-sia. Hal pengembangan sistem yang paling berguna adalah proses dokumentasi sistem dimana didalamnya tersimpan semua hal yang penting yang dapat menjadi tolak ukur pengembangan sistem pada masa mendatang termasuk didalamnya adalah kamus pengetahuan masalah yang diselesaikan.

II.2. Representasi Pengetahuan

Pengetahuan yang didapatkan pada proses akuisisi pengetahuan tidak bisa diaplikasi begitu saja dalam sistem pakar. Pengetahuan harus direpresentasikan dalam format tertentu dan akan dihimpun dalam suatu basis pengetahuan.

II.2.1 Model Representasi Pengetahuan

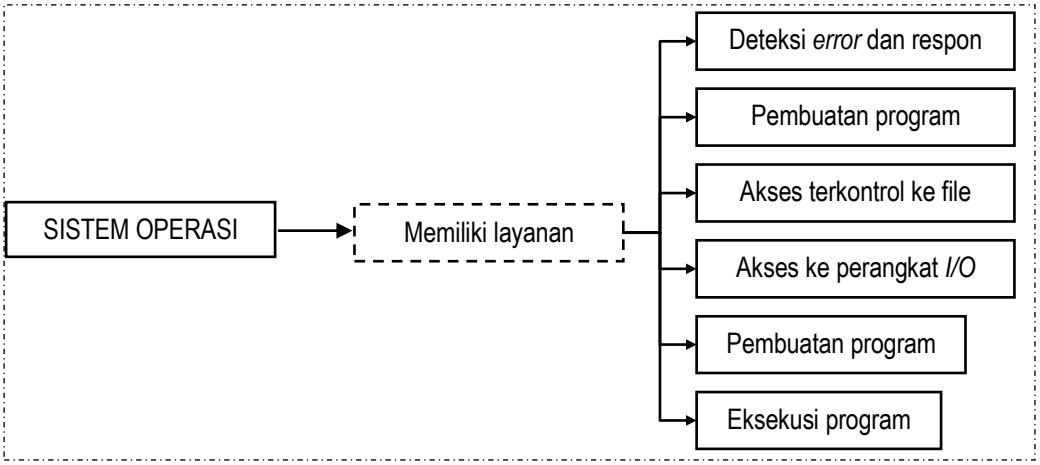
Representasi pengetahuan dimaksudkan untuk mengorganisasikan pengetahuan dalam bentuk tertentu. Untuk membuat sistem pakar yang efektif harus dipilih representasi pengetahuan yang tepat. Beberapa model representasi pengetahuan yang penting:

1. Jaringan Semantik (*Semantic Nets*)
2. Bingkai (*Frames*)
3. Kaidah Produksi (*Production Rule*)

II.2.2. Jaringan Semantik (*Semantic Nets*)

Jaringan semantik adalah teknik representasi pengetahuan yang digunakan untuk informasi proposional. Yang dimaksud dengan informasi proporsional adalah pernyataan yang mempunyai nilai benar atau salah. Sebagai contoh, sebuah bujur sangkar mempunyai empat sisi. Informasi proporsional merupakan bahasa deklaratif karena menyatakan fakta.

Representasi jaringan semantik merupakan penggambaran grafis dari pengetahuan yang memperlihatkan hubungan hirarkis dari obyek-obyek. Komponen dasar untuk merepresentasikan pengetahuan dalam bentuk jaringan semantik adalah simpul (*node*) dan penghubung (*link*). Objek direpresentasikan oleh simpul. Hubungan antar obyek-obyek dinyatakan oleh penghubung yang di beri label untuk menyatakan hubungan yang direpresentasikan. Contoh jaringan semantik terlihat pada gambar II.4.



Gambar II.4 Contoh Jaringan Semantik

Sumber : Andi, 2009:32

II.2.3. Bingkai (*Frames*)

Bingkai berupa kumpulan slot-slot yang berisi atribut untuk mendeskripsikan pengetahuan. Pengetahuan yang termuat dalam slot dapat berupa

kejadian, lokasi, situasi ataupun elemen-elemen lainnya. Bingkai digunakan untuk representasi pengetahuan deklaratif.

Bingkai memuat deskripsi sebuah obyek dengan menggunakan tabulasi informasi yang berhubungan dengan obyek. Jadi bingkai mengelompokkan atribut sebuah obyek. Dengan demikian bingkai membantu menirukan cara seorang mengorganisasikan informasi tentang sebuah obyek menjadi kumpulan data. Contoh bingkai terlihat pada tabel II.3.

Tabel II.3 Contoh Bingkai

Slot	*Fillers*
Nama	Pajero Sport
Spesialisasi	Jenis kendaraan beroda empat
Pembuat	Mitsubishi
Isi silinder	7,5 L
Pelumas	Diesel

Sumber : Kusrini, 2009 :27

II.2.4. Kaidah Produksi (*Production Rule*)

Kaidah menyediakan cara formal untuk merepresentasikan rekomendasi, arahan atau strategi. Kaidah produksi dituliskan dalam bentuk jika-maka (*if-then*). Kaidah *if-then* menghubungkan antesenden (*antecedent*) dengan konsekuensi yang diakibatkannya(Sri Hartati dan Sari Iswanti, 2009). Berbagai struktur kaidah *if-then* yang menghubungkan obyek atau atribut sebagai berikut:

IF Premis *THEN* konklusi

IF masukan *THEN* keluaran

IF kondisi *THEN* tindakan

IF antesenden *THEN* konsekuen

IF Data *THEN* hasil

IF Tindakan *THEN* tujuan

Premis mengacu pada fakta yang harus benar sebelum konklusi tertentu dapat diperoleh. Masukan mengacu pada data yang harus tersedia sebelum keluaran dapat diperoleh. Kondisi mengacu pada keadaan yang harus berlaku sebelum tindakan dapat diambil. Data mengacu pada informasi yang harus tersedia sehingga sebuah hasil dapat diperoleh. Tindakan mengacu pada kegiatan yang harus dilakukan sebelum hasil dapat diharapkan. Beberapa kaidah yang diberikan dalam contoh di bawah ini menunjukkan beberapa bentuk pengetahuan peluang diekspresikan ke dalam bentuk format kaidah produksi yaitu :

Kaidah 1 (cf = 0.8) :

IF beberapa tombol keyboard tidak berfungsi (cf = 0,65)
AND keyboard tidak merespon sama sekali (cf = 0.7)
AND keyboard bisa bekerja setelah dibersihkan (cf = 0.3)
AND hubungkan ke kompoter lain tetap tidak respon (cf = 0.8)
THEN Keyboard Kotor
Cf konklusi = 0.24

Clouse adalah semacam kalimat yang terdiri dari subyek, kata kerja dan objek yang menyatakan beberapa fakta. Pada setiap *clouse* terdapat satu *clouse If* dan satu *clouse Then*. Bagian kaidah *if* mungkin berisi lebih dari satu premis, dan masing-masing berisi *clouse*. Setiap kaidah yang mempunyai lebih dari satu premis disebut *compound clouse* dan dihubungkan oleh kata penghubung *AND* atau *OR*.

Hal yang tidak boleh dilupakan dalam kaidah produksi adalah bahwa kaidah produksi menampilkan se-kelumit pengetahuan individual. Hal ini biasanya dihubungkan dengan banyak kaidah. Kaidah tersebut dikaitkan secara bersama-sama untuk membentuk garis penalaran.

II.3 Faktor Ketidakpastian

Suatu pernyataan kondisi maupun aksi tidak selalu memberikan derajat kepastian yang penuh, akan tetapi kadang-kadang tergantung pada suatu faktor tertentu, karena dalam kenyataannya para pakar seringkali berurusan dengan fakta-fakta yang tidak menentu dan tidak pasti. Untuk itu sistem pakar juga harus dapat menangani masalah kekurangpastian dan ketidakpastian. Teknik yang sudah digunakan untuk menangani hal tersebut diantaranya adalah nilai pendekatan Bayes (*Bayesian Approach*), faktor kepastian (*Certainty Factor*), teori *Dempstre-Shafer* (M. Arhami, 2010).

II.3.1. Pendekatan Bayes (*Bayesian Approach*)

Probabilitas Bayes merupakan salah satu cara yang baik untuk mengatasi ketidakpastian data dengan menggunakan formula bayes yang dinyatakan dengan rumus : $P(H|E)=\frac{P(E|H).P(H)}{P(E)}$

Keterangan :

P(H | E) : probabilitas hipotesis H jika diberikan

Evidence E

P(E | H) : probabilitas munculnya *evidence* apapun

P(E) : probabilitas *evidence* E

Contoh kasus penyakit gagal ginjal akut : (Sumber : Rahayu,2013)

Aulia melakukan diagnosa dengan menjawab pertanyaan sesuai dengan gejala berikut :

G1 = 0.4 = P(E|H1) G5 = 0.2 = P(E|H5) G9 = 0.4 = P(E|H9)
G2 = 0.2 = P(E|H2) G6 = 0.2 = P(E|H6)
G3 = 0.4 = P(E|H3) G7 = 0.2 = P(E|H7)
G4 = 0.6 = P(E|H4) G8 = 0.4 = P(E|H8)

Kemudian mencari nilai semesta dengan menjumlahkan dari hipotesa di atas :

$$\sum^{9}_{k=1} = G1 + G2 + G3 + G4 + G5 + G6 + G7 + G8 + G9$$
$$= 0.4+ 0.2 + 0.4 + 0.6 + 0.2 + 0.2 + 0.2 + 0.4 + 0.4$$
$$= 3$$

Setelah hasil penjumlahan di atas diketahui, maka didapatlah rumus untuk menghitung nilai semesta adalah sebagai berikut :

$$P(H\,1) = \frac{H\,1}{\sum^{9}_{k=1}} = \frac{0.4}{3} = 0.133333$$
$$P(H\,2) = \frac{H\,2}{\sum^{9}_{k=1}} = \frac{0.2}{3} = 0.066667$$
$$P(H\,3) = \frac{H\,3}{\sum^{9}_{k=1}} = \frac{0.4}{3} = 0.133333$$
$$P(H\,4) = \frac{H\,4}{\sum^{9}_{k=1}} = \frac{0.6}{3} = 0.200000$$
$$P(H\,5) = \frac{H\,5}{\sum^{9}_{k=1}} = \frac{0.2}{3} = 0.066667$$
$$P(H\,6) = \frac{H\,6}{\sum^{9}_{k=1}} = \frac{0.2}{3} = 0.066667$$
$$P(H\,7) = \frac{H\,7}{\sum^{9}_{k=1}} = \frac{0.2}{3} = 0.066667$$
$$P(H\,8) = \frac{H\,8}{\sum^{9}_{k=1}} = \frac{0.4}{3} = 0.133333$$
$$P(H\,9) = \frac{H\,9}{\sum^{9}_{k=1}} = \frac{0.4}{3} = 0.133333$$

Setelah nilai P(Hi) diketahui, probabilitas hipotesis H tanpa memandang *evidence* apapun, maka langkah selanjutnya adalah :

$\sum^{9}_{k=1}$ = P (Hi)* P(E|Hi-n)
= P(H1) * P(E|H1) + P(H2) * P(E|H2) + P(H3) * P(E|H3) + P(H4) * P(E|H4) +
P(H5) * P(E|H5) + P(H6) * P(E|H6) + P(H7) * P(E|H7) + P(H8) * P(E|H8) + P(H9) * P(E|H9)
=(0.133333*0.4) + (0.066667*0.2) + (0.133333*0.4) + (0.200000*0.6) + (0.066667 *0.2) + (0.066667 *0.2) + (0.066667 *0.2) (0.133333*0.4) + (0.133333*0.4)
= 0.053333+0.013333+0.053333+0.12+0.013333+0.013333+0.013333+ 0.053333+0.053333 = 0.386667

Langkah selanjutnya ialah mencari nilai P(Hi|E) atau probabilitas hipotesis Hi benar jika diberikan *evidence* E

P(H1|E) = 0.133333*0.4 = 0.137931
0.386667
P(H2|E) = 0.034483*0.2 = 0.034483
0.386667
P(H3|E) = 0.133333*0.4 = 0.137931
0.386667
P(H4|E) = 0.310345*0.6 = 0.310345
0.386667
P (H9| E) = 0.133333*0.4 = 0.137931
0.386667

P(H5|E) = 0.034483*0.2= 0.034483
0.386667
P(H6|E) = 0.034483*0.2= 0.034483
0.386667
P(H7| E) = 0.034483*0.2= 0.034483
0.386667
P(H8|E) = 0.133333*0.4 = 0.137931
0.386667

Setelah seluruh nilai P(Hi|E) diketahui, maka jumlahkan seluruh nilai bayesnya dengan rumus sebagai berikut :

$\sum^{9}_{k=1}$ = Bayes 1 + Bayes 2 + Bayes 3 + Bayes 4 + Bayes 5 + Bayes 6 + Bayes7+ Bayes 8 + Bayes 9
= (0.137931*0.4) + (0.034483*0.2) + (0.137931*0.4) + (0.310345*0.6) + (0.034483*0.2) + (0.034483*0.2) + (0.034483*0.2) + (0.137931*0.4) + (0.137931*0.4)
= 0.434488= 43, 44828

II.4. Kerusakan Mobil

Terkadang mobil juga menunjukkan beberapa gejala yang merupakan petunjuk bahwa mobil dalam keadaan kurang baik. Bila mobil mulai menunjukkan adanya penurunan performa, berarti mesin dalam mobil tidak dapat bekerja secara maksimal. Ada beberapa penyebab yang bisa menyebabkan terjadinya kerusakan pada mesin mobil anda. Bila penyebab kerusakan pada

mesin mobil tidak segera ditangani dengan baik, bisa menyebabkan kerusakan total yang lebih parah lagi nantinya.

II.4.1. Masalah Utama Mobil dan Penyebabnya

Gejala-gejala kerusakan mobil yang umum terjadi dapat dirincikan sebagai berikut: (http://www.tempo.co)

1. Mesin sulit diaktifkan pada pagi hari

   Ada beberapa faktor yang menjadi penyebab mesin mobil sulit sekali dihidupkan di saat pagi hari atau mesin dingin. Beberapa penyebabnya, antara lain tegangan arus listrik aki yang kurang atau bahkan telah habis, terminal aki yang berkarat atau kotor dan kendor. Penyebab lainnya adalah injektor bahan bakar bermasalah, jarak antara kepala busi dan sumbu busi terlalu renggang, kompresi mesin yang terlalu rendah, regulator tekanan bahan bakar rusak.

2. Mesin sulit diaktifkan saat panas

   Kondisi ini merupakan kebalikan dari kondisi nomor satu, yaitu mesin sulit sekali dinyalakan setelah mobil berjalan jauh dan kemudian berhenti atau mesin dimatikan. Penyebab kondisi ini adalah kompresi silinder yang rendah, choke bermasalah, injektor kotor, atau saringan udara yang kotor dan tersumbat.

3. Mesin tersendat saat pedal gas diinjak untuk meningkatkan akselerasi

   Kondisi seperti ini kerap terjadi di mobil yang telah berusia di atas lima tahun. Beberapa penyebab permasalahan ini adalah busi yang telah aus atau jarak antara kepala dan sumbu busi terlalu renggang, vacuum di karburator, throttle

body, intake manifold atau di bagian selang vacuum yang bocor. Penyebab lainnya adalah injektor bahan bakar yang kotor atau tersumbat.

4. Tenaga mesin kurang bertenaga tidak seperti biasanya

   Persoalan lain yang sering dikeluhkan oleh para pemilik mobil adalah merosotnya tenaga yang dihasilkan oleh mesin. Mobil kurang bertenaga dan tak berdaya seperti sebelumnya.

   Ada beberapa faktor yang menyebabkan hal itu terjadi, di antaranya pemasangan timing belt yang tidak tepat, saringan bahan bakar yang kotor sehingga tersumbat, fuel pump bermasalah, regulator tekanan bahan bakar rusak, tingkat kompresi satu atau beberapa silinder menurun, dan kerenggangan kepala dan sumbu busi terlalu jauh.

   Penyebab lainnya adalah vacuum di karburator, throttle body, inlet manifold, serta selang yang bocor. Bisa juga karena injektor bermasalah, rem yang mengunci sehingga mobil dalam posisi terus mengerem, atau karena kopling slip sehingga tidak ada perubahan posisi gigi meskipun putaran mesin tinggi.

5. Mesin hidup mati, hidup mati

   Beberapa pemilik mobil mengungkapkan pernah mengalami mesin mobilnya langsung hidup saat kunci kontak diputar ke on. Namun, hal itu tidak berlangsung lama, karena mesin mati. Anehnya kondisi seperti itu berlangsung berulang kali.

   Ada beberapa faktor yang menjadi penyebabnya, di antaranya sistem kelistrikan (mulai dari alternator, aki, hingga kabel-kabel yang kendur atau putus), serta kemungkinan rangkaian pengapian yang bermasalah.

Penyebab lainnya adalah, vacuum pada karburator, throttle body, inlet manifold, atau selang-selang vacuum yang bocor. Bisa juga injektor yang bermasalah karena kotor atau tersumbat.

6. Lampu indikator oli menyala terus meski mobil telah berjalan

   Masalah ini juga sering terjadi dan kerap tidak disadari oleh para pemilik mobil atau pengemudi. Umumnya mereka hanya percaya bahwa mobilnya telah rutin ganti oli sehingga lampu indikator luput dari perhatian.

   Lampu yang terus menyala meski mesin telah cukup lama aktif bisa dikarenakan beberapa faktor, di antaranya jenis oli yang tidak sesuai dengan spesifikasi mobil atau tidak sesuai rekomendasi dari pabrikan. Selain itu juga dikarenakan jumlah oli yang tidak sesuai dengan standar volume, sensor dan katup peranti tekanan oli rusak, atau saringan oli yang sudah aus.

7. Mesin mendesis

   Beberapa orang pernah mengeluhkan mesin mobil mereka mengeluarkan suara mendesis meski sudah lama dipanaskan. Penyebab masalah ini adalah, inlet manifold atau gasket throttle body bocor, exhaust manifold bocor, selang vacuum bocor, serta gasket di kepala silinder rusak sehingga bocor atau robek.

8. Mesin menggelitik

   Mesin knocking atau menggelitik juga kerap dikeluhkan oleh para pemilik mobil, terutama di saat mereka berusaha untuk menambah kecepatan akselerasi atau di saat mobil melibas jalanan menanjak.

   Ada beberapa penyebab. Faktor penyebab utama dan paling sering adalah spesifikasi bahan bakar yang jelek, tingkat Research Octane Number (RON) bahan bakar yang tidak sesuai. Walhasil, timing pengapian tidak tepat.

Istilah bengkelnya, karena ada premature ignition yaitu proses pembakaran di ruang bakar antara bahan bakar dan udara dengan pantikan api tidak sempurna. Sehingga ada ledakan prematur, itulah knocking.

Penyebab lainnya adalah, injektor atau karburator mobil yang bermasalah, jarak antara kepala dan sumbu busi terlalu renggang, vacuum karburator, throttle body, inlet manifold, dan selang yang bocor.

II.4.2. Gejala Kerusakan Mobil

Diusia dan waktu yang tertentu mobil akan mengalami penurunan kondisi. Hal itu tentu berkaitan dengan tingkat keausan komponen baik akibat pemakaian, kondisi jalan maupun usia komponen itu sendiri. Namun dengan jumlah yang sangat banyak, sulit rasanya jika kita harus mengecek kondisi tiap komponen. Beberapa penyakit pada mobkas memang bisa dideteksi langsung dengan mata. Namun banyak juga gejala kerusakan yang baru muncul ketika melakukan test drive. Makanya, jangan ragu untuk meminta test drive mobil bekas yang sedang diincar.

Dengan melakukan test drive, maka dapat diketahui kerusakan apa saja yang bisa muncul. Ada beberapa kerusakan umum yang terjadi di kebanyakan mobil bekas. Dengan mengenali gejala penyakit umum ini, tentu bisa dijadikan bahan pertimbangan sekaligus persiapan dana untuk perbaikan jika memang ingin membelinya.

1. Gejala: Setir bergetar pada kecepatan tertentu.

   Umumnya, penyakit ini disebabkan roda depan sudah tidak *balance*. Jika getaran itu juga dirasakan di bangku *driver*, kemungkinan roda belakang juga

tidak *balance*. Membalans roda dengan biaya sekitar Rp 30.000, getaran itu bisa hilang.

2. Gejala: Ketika melindas gundukan, mobil memantul lebih dari dua kali. Kondisi ini lebih sulit dideteksi jika kecepatan melindas gundukan terlalu pelan. Coba gunakan kecepatan sedikit lebih tinggi untuk melindas gundukan berikutnya. Umumnya penyakit ini disebabkan keausan atau kerusakan sokbreker. Kerusakan ini terjadi karena umur sokbreker. Anda bisa memeriksa kondisi fisiknya setelah test drive. Dengan mengganti sokbreker baru dengan kisaran harga Rp 1-2 juta, masalah ini bisa tuntas.
3. Gejala: Sejak kecepatan rendah, mobil terasa bergoyang

   Jika Anda merasakan goyang bodi selaras dengan setir, bisa jadi salah satu pelek roda depan sudah peyang. Namun jika pelek roda belakang yang peyang, biasanya goyangan begitu terasa jika duduk di bangku belakang.

   Untuk menyelesaikan masalah ini, Anda bisa mereparasi pelek dengan biaya sekitar Rp 150 ribu per piece. Usai test drive, tak ada salahnya Anda mengecek juga kondisi pelek ban serep. Penyebab lainnya, salah satu roda tidak terpasang dengan baik akibat pemasangan baut roda yang kurang kencang.
4. Gejala: Ketika jalan lurus, posisi setir tidak lurus

   Kemungkinan setir pernah dilepas dan ketika pemasangan kembali, posisi roda kemudi tidak lurus. Sehingga tidak selaras dengan arah ban. Kemungkinan lain adalah masalah sudut-sudut keselarasan roda. Dengan melakukan wheel alignment (spooring) di bengkel, masalah ini bisa dihilangkan.

Tapi jika Anda mengajak teknisi, sebaiknya cek pula komponen sistem kemudi. Terutama jika gejala ini muncul setelah Anda melindas gundukan atau lubang jalan. Indikator lainnya yakni kondisi keausan tapak ban yang tidak merata. Jika ini yang terjadi, perbaikan sistem kemudi bisa menelan biaya besar.

5. Gejala: Ketika setir dilepas, mobil cenderung mengarah ke salah satu sisi

   Jika Anda berada di jalur kiri dan mobil cenderung ke kiri, biasanya karena permukaan jalan yang memang miring ke kiri. Untuk memastikannya, coba cek tekanan angin semua ban. Sekaligus memeriksa apakah kedua roda depan menggunakan ukuran yang sama. Jika berbeda ukuran ban, masalah bisa selesai dengan menyamakannya.

   Namun jika terjadi ketika Anda melaju di permukaan jalan yang rata, artinya perlu melakukan wheel alignment di bengkel. Sementara kemungkinan terburuk adalah penyakit ini disebabkan konstruksi sistem kemudi mengalami kerusakan.

   Coba periksa apakah ada batang kemudi di kolong mobil yang melengkung. Biasanya hal ini terjadi karena pernah terbentur benda keras atau menggunakan batang itu sebagai kaitan ketika ditarik mobil lain. Kondisi ini membahayakan untuk dikemudikan.

6. Gejala: Muncul suara berdengung dari roda ketika melaju di jalan aspal mulus

   Suara dengung ini bisa muncul akibat pengunaan ban baik standar maupun aftermarket dengan motif kembang yang cukup besar atau untuk keperluan off-road. Penyebab lainnya adalah rotasi ban yang sudah lama tidak dilakukan. Coba periksa kondisi ban depan dengan meraba bagian pinggir tapaknya

secara perlahan dalam dua arah. Namun hati-hati jika ada benang baja yang sudah keluar atau benda tajam lain. Ketika diraba ke satu arah terasa mulus, tapi saat berbalik arah Anda mendapati kembang ban lebih tajam atau kasar, kondisi ini bisa menimbulkan dengung.

Hal ini umumnya muncul akibat pemakaian. Begitu pula desain ban yang umumnya menggunakan kembang yang cukup tebal di bagian pinggir tapak yang berguna untuk membuat jalur air ketika hujan. Solusinya adalah dengan merotasi ban. (http://www.tempo.co)

II.5. Pajero Sport

Mitsubishi Pajero adalah SUV yang dibuat oleh Mitsubishi Motors dikenal sebagai Mitsubishi Montero di Amerika (kecuali Brazil) dan Spanyol, dan sebagai Mitsubishi Shogun di Inggris.

II.5.1. Tipe- tipe Pajero Sport dan Spesifikasi

Berikut ini beberapa jenis Pajero Sport adalah sebagai berikut : (http://sardanagroup.co.id/product/pc/pajero-sport).

1. Pajero Sport Dakar 4x4 2,5L 5AT HP

   Spesifikasi :

Tabel II.4 Spesifikasi Pajero Sport Dakar 4x4 2,5L 5AT HP

ITEM		DESCRIPTION
DIMENSION & WEIGHT		
Overall Length	mm	4,695
Overall Width	mm	1,815
Overall Heigth	mm	1,840
Wheel Base	mm	2,800
Tread Front	mm	1,520
Tread Rear	mm	1,515
Front Overhang	mm	795
Rear Overhang	mm	1,100
Ground Clearance	mm	215
Gross Weight Vehicle	Kg	2,710
Curb Weight	Kg	2,075
Seating Capacity		7
ENGINE		
Engine Type		2.5L DOHC Commonrail Turbocharged and Intercooled, 4 Cylinder in-line (4D56) (HIGH POWER)
Displacement	cc	2,477
Bore x Stroke	mm	91.1 x 95.0
Compression Rate		16.5 : 1
Max. Horse Power	PS/rpm	178/4,000 (131kW)
Max. Torque	Kgm/rpm	35.7/1,800 - 3,500 (350Nm)
Fuel Type		Diesel
Fuel System		Common Rail (Low Pressure) Direct Injection
Fuel Tank Capacity	Litre	70
PERFORMANCE		
Min. Turning Radius	m	5.6
Approach Angle	°	36
Ramp Break Over Angle	°	23
Departure Angle	°	24
TRANSMISSION		
Type		V5A5A (5 A/T)
		INVECS II with Sport Mode
Gear Ratio	1st	3.789
	2nd	2.057
	3rd	1.421
	4th	1.000
	5th	0.731
	Reverse	3.865
Final Gear Ratio		3.917
4WD System		Super Select 4WD
LSD		Available (Hybrid Type)
SUSPENSION		
Front		Double Independent Wishbone, Coil Spring with Stabilizer
Rear		3 Link Coil Spring with Stabilizer
BRAKE		
Brake	Front	Ventilated Disc 16"
	Rear	Rear 16" drum in disc
Vacum Brake Booster		Available
ABS		Available
EBD		Available
TIRE		
Size	Front	265 / 65 R17
	Rear	265 / 65 R17
Wheel Disc		17" x 7.5J Alloy Wheel
Spare Wheel		17" x 7.5J Alloy Wheel
STEERING		
Type		Rack & Pinion with Power Steering
Paddle Shift		Available
Audio Switch		Available
Steering Wheel		3 Spoke Leather Wrapped
Tilt Steering		Available

SAFETY		
Front SRS Airbags	Driver	Available
	Passenger	Available
Seatbelt	Front Row	3P ELR with Adjustable Anchor & Pretensioner x 2
	2nd Row	3P ELR x 2 and 2P Lapbelt x 1
	3rd Row	3P ELR x 2
Child Proof Lock		Available
Side Impact Beams (Fr & Rr)		Available
Laminated Green Glass		Available
AUDIO SYSTEM		
Type		Wide 2 DIN Multimedia with NAVI, Mitsubishi Power Sound System (High Power Amp. 420W total max.) + Rear Camera
Audio Jack		Available
Speaker		8
Antenna		Roof Mounted
Instrument Panel (2-Tone)		Black - Beige
Seat Material		Leather (Dark Brown)
Height Seat Adjuster		Available (Manual)
EXTERIOR		
Front & Rear Bumper - Color Keyed		Available
Radiator Front Grille		Horizontal Type (Chrome)
License Plate Garnish		Full Chrome
License Plate Frame		Available
Rear Spoiler		Available
Chrome Cover Fog Lamp		Available
Muffler Cutter		Available
ELECTRICAL		
H.I.D. with Auto Leveling Device		Available
Washer Headlamp		Available
Auto Up Power Window		Available
Power Tilt and Sliding Sunroof		Available
Power Door Lock		Available
Multi Information Display		Available
Auto Lock & Auto Folding		Available
Air Filter		Available

2. Pajero Sport Dakar 4x2 2,5L 5AT HP

Spesifikasi :

Tabel II.5 Spesifikasi Pajero Sport 4x2 2,5L 5AT HP

ITEM		DESCRIPTION
DIMENSION & WEIGHT		
Overall Length	mm	4,695
Overall Width	mm	1,815
Overall Heigth	mm	1,840
Wheel Base	mm	2,800
Tread Front	mm	1,520
Tread Rear	mm	1,515
Front Overhang	mm	795
Rear Overhang	mm	1,100
Ground Clearance	mm	215
Gross Weight Vehicle	Kg	2,600
Curb Weight	Kg	1,965
Seating Capacity		7
ENGINE		
Engine Type		2.5L DOHC Commonrail Turbocharged and Intercooled, 4 Cylinder in-line (4D56) (HIGH POWER)
Displacement	cc	2,477
Bore x Stroke	mm	91.1 x 95.0
Compression Rate		16.5 : 1
Max. Horse Power	PS/rpm	178/4,000 (131kW)
Max. Torque	Kgm/rpm	35.7/1,800 - 3,500 (350Nm)
Fuel Type		Diesel
Fuel System		Common Rail (Low Pressure) Direct Injection
Fuel Tank Capacity	Litre	70
PERFORMANCE		
Min. Turning Radius	m	5.6
Approach Angle	°	36
Ramp Break Over Angle	°	23
Departure Angle	°	24
TRANSMISSION		
Type		R5A5A (5 A/T)
		INVECS II with Sport Mode
Gear Ratio	1st	3.789
	2nd	2.057
	3rd	1.421
	4th	1.000
	5th	0.731
	Reverse	3.865
Final Gear Ratio		3.917
4WD System		-
LSD		Available (Hybrid Type)

SUSPENSION		
Front		Double Independent Wishbone, Coil Spring with Stabilizer
Rear		3 Link Coil Spring with Stabilizer
BRAKE		
Brake	Front	Ventilated Disc 16"
	Rear	Rear 16" drum in disc
Vacum Brake Booster		Available
ABS		Available
EBD		Available
TIRE		
Size	Front	265 / 65 R17
	Rear	265 / 65 R17
Wheel Disc		17" x 7.5J Alloy Wheel
Spare Wheel		17" x 7.5J Alloy Wheel
STEERING		
Type		Rack & Pinion with Power Steering
Paddle Shift		Available
Audio Switch		Available
Steering Wheel		3 Spoke Leather Wrapped
Tilt Steering		Available
SAFETY		
Front SRS Airbags	Driver	Available
	Passenger	Available
	Front Row	3P ELR with Adjustable Anchor & Pretensioner x 2
SUSPENSION		
Front		Double Independent Wishbone, Coil Spring with Stabilizer
Rear		3 Link Coil Spring with Stabilizer
BRAKE		
Brake	Front	Ventilated Disc 16"
	Rear	Rear 16" drum in disc
Vacum Brake Booster		Available
ABS		Available
EBD		Available
TIRE		
Size	Front	265 / 65 R17
	Rear	265 / 65 R17
Wheel Disc		17" x 7.5J Alloy Wheel
Spare Wheel		17" x 7.5J Alloy Wheel
STEERING		
Type		Rack & Pinion with Power Steering
Paddle Shift		Available
Audio Switch		Available
Steering Wheel		3 Spoke Leather Wrapped
Tilt Steering		Available
SAFETY		
Front SRS Airbags	Driver	Available
	Passenger	Available
	Front Row	3P ELR with Adjustable Anchor & Pretensioner x 2
Seatbelt	2nd Row	3P ELR x 2 and 2P Lapbelt x 1
	3rd Row	3P ELR x 2
Child Proof Lock		Available
Side Impact Beams (Fr & Rr)		Available
Laminated Green Glass		Available
AUDIO SYSTEM		
Type		Wide 2 DIN Multimedia with NAVI, Mitsubishi Power Sound System (High Power Amp. 420W total max.) + Rear Camera
Audio Jack		Available
Speaker		8
Antenna		Roof Mounted
EQUIPMENT		

INTERIOR	
Instrument Panel (2-Tone)	Black - Beige
Seat Material	Leather (Black)
Height Seat Adjuster	Available (Manual)
EXTERIOR	
Front & Rear Bumper - Color Keyed	Available
Radiator Front Grille	Horizontal Type (Chrome)
License Plate Garnish	Full Chrome
License Plate Frame	Available
Rear Spoiler	Available
Chrome Cover Fog Lamp	Available
Muffler Cutter	Available
ELECTRICAL	
H.I.D. with Auto Leveling Device	Available
Washer Headlamp	Available
Auto Up Power Window	Available
Power Tilt and Sliding Sunroof	Available
Power Door Lock	Available
Multi Information Display	Available
Auto Lock & Auto Folding	Available
Air Filter	Available

3. Pajero Sport Exceed 4x2 2,5L AT

Spesifikasi :

tabel II.6 Spesifikasi Pajero Exceed 4x2 2,5L 5AT HP

ITEM		DESCRIPTION
DIMENSION & WEIGHT		
Overall Length	mm	4,695
Overall Width	mm	1,815
Overall Heigth	mm	1,840
Wheel Base	mm	2,800
Tread Front	mm	1,520
Tread Rear	mm	1,515
Front Overhang	mm	795
Rear Overhang	mm	1,100
Ground Clearance	mm	215
Gross Weight Vehicle	Kg	2,650
Curb Weight	Kg	1,925
Seating Capacity		7
ENGINE		
Engine Type		2.5L DOHC Commonrail Turbocharged and Intercooled, 4 Cylinder in-line (4D56)
Displacement	cc	2,477
Bore x Stroke	mm	91.1 x 95.0
Compression Rate		17 : 1
Max. Horse Power	PS/rpm	136/3,500 (100kW)
Max. Torque	Kgm/rpm	32/2,000 (314Nm)
Fuel Type		Diesel
Fuel System		Common Rail (Low Pressure) Direct Injection
Fuel Tank Capacity	Litre	70
PERFORMANCE		
Min. Turning Radius	m	5.6
Approach Angle	°	36
Ramp Break Over Angle	°	23
Departure Angle	°	24
TRANSMISSION		

Type		R4A6A (4 A/T)
		INVECS II with Sport Mode
Gear Ratio	1st	2.842
	2nd	1.459
	3rd	1.000
	4th	0.731
	5th	-
	Reverse	2.720
Final Gear Ratio		4.100
4WD System		-
LSD		-
SUSPENSION		
Front		Double Independent Wishbone, Coil Spring with Stabilizer
Rear		3 Link Coil Spring with Stabilizer
BRAKE		
Brake	Front	Ventilated Disc 16"
	Rear	Leading & Trailing Drum 11.6"
Vacum Brake Booster		Available
ABS		Available
EBD		Available
TIRE		

Size	Front	265 / 65 R17
	Rear	265 / 65 R17
Wheel Disc		17" x 7.5J Alloy Wheel
Spare Wheel		17" x 7.5J Alloy Wheel
STEERING		
Type		Rack & Pinion with Power Steering
Paddle Shift		Not Available
Audio Switch		Available
Steering Wheel		3 Spoke Leather Wrapped
Tilt Steering		Available
SAFETY		
Front SRS Airbags	Driver	Available
	Passenger	Available
Seatbelt	Front Row	3P ELR with Adjustable Anchor & Pretensioner x 2
	2nd Row	3P ELR x 2 and 2P Lapbelt x 1
	3rd Row	3P ELR x 2
Child Proof Lock		Available
Side Impact Beams (Fr & Rr)		Available
Laminated Green Glass		Available
AUDIO SYSTEM		
Type		Wide 2 DIN Multimedia with NAVI, + Rear Camera
Audio Jack		Available
Speaker		6
Antenna		Roof Mounted
EQUIPMENT		
INTERIOR		
Instrument Panel (2-Tone)		Black - Beige
Seat Material		Leather (Beige)
Height Seat Adjuster		Available (Manual)
EXTERIOR		
Front & Rear Bumper - Color Keyed		Available
Radiator Front Grille		Horizontal Type (Chrome)
License Plate Garnish		Color Keyed and Chrome Combination
License Plate Frame		Not Available
Rear Spoiler		Not Available
Chrome Cover Fog Lamp		Not Available
Muffler Cutter		Not Available
ELECTRICAL		

H.I.D. with Auto Leveling Device	Not Available
Washer Headlamp	Not Available
Auto Up Power Window	Available
Power Tilt and Sliding Sunroof	Not Available
Power Door Lock	Available
Multi Information Display	Available
Auto Lock & Auto Folding	Not Available
Air Filter	Available

II.6. *Forward Chaining*

Metode *forward chaining* adalah sebuah metode penelusuran pohon pencarian yang mencari dari akar pohon menelusuri cabang pohon menuju ke daun pohon. Inferensi majemuk yang menghubungkan permasalahan dengan solusinya disebut sebagai "chain" (rantai). Rantai yang yang dilalui dari masalah ke solusi disebut "forward chain" (penalaran dari fakta menuju konklusi berdasarkan fakta-fakta tersebut).

Contoh : Misalkan memiliki kaidah dari modus ponen:

$p \rightarrow q$

p

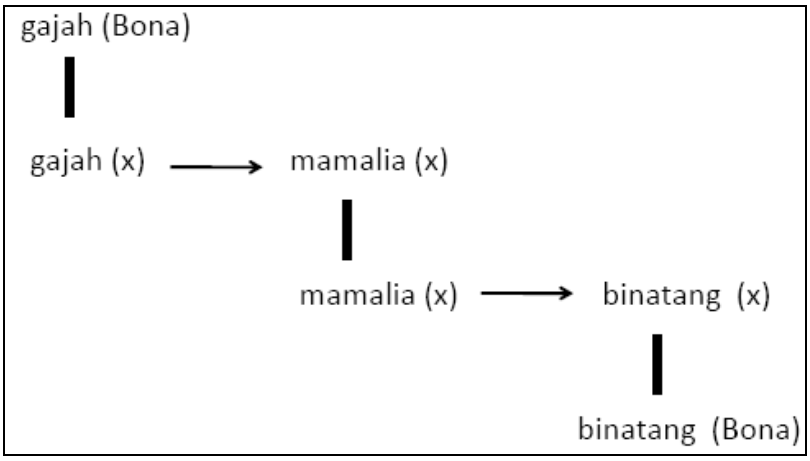
$\therefore q$

dalam bentuk :

gajah (x) → mamalia (x)

mamalia (x) → binatang (x)

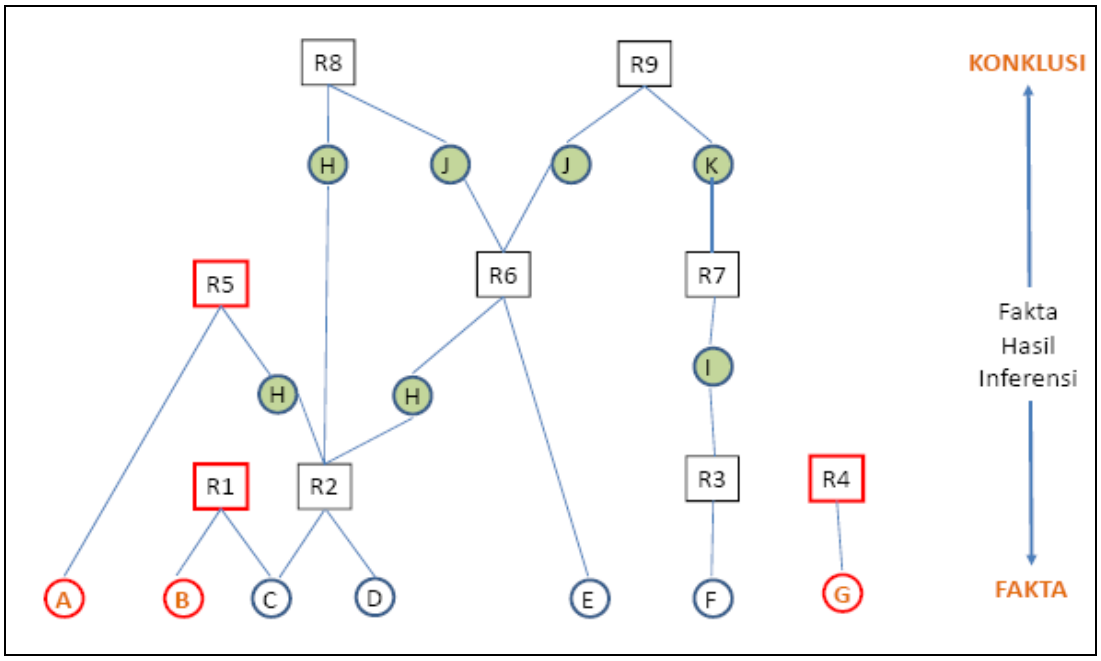
Kaidah ini dapat digunakan dalam rantai sebab-akibat dari inferensi forward yang menarik kesimpulan bahwa Bona adalah binatang, jika diketahui Bona adalah gajah.



Gambar II.5 Ilustrasi Contoh Kaidah

Sumber : Kusrini, 2009:51

Seperti terlihat pada gambar di atas, Bona adalah gajah dan gajah termasuk mamalia, dimana mamalia adalah binatang. Hal ini berarti Bona adalah binatang.



Gambar II.6 Ilustrasi *Forward Chaining*

Sumber : Kusrini, 2009:53

Contoh :

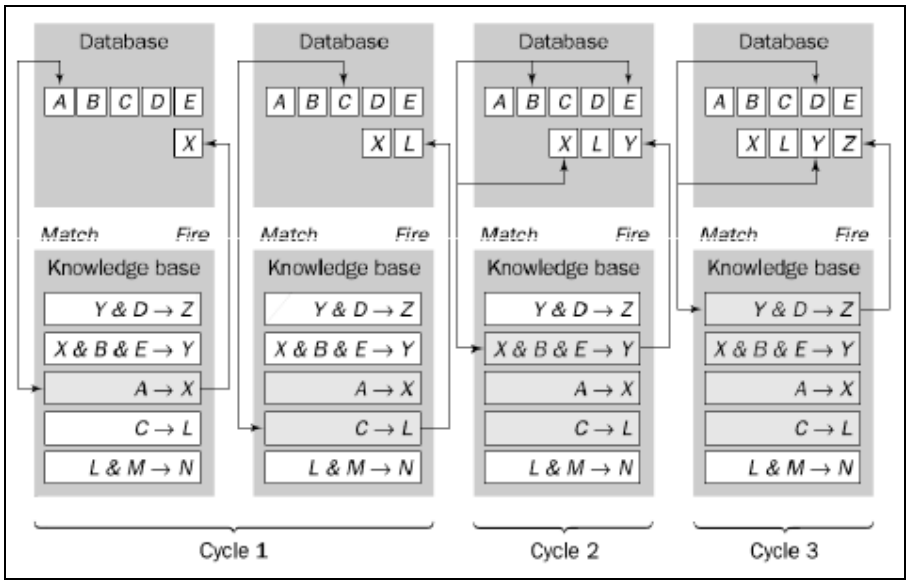
Rule 1 : Y & D → Z

Rule 2 : X & B & E → Y

Rule 3 : A → X

Rule 4 : C → L

Rule 5 : L & M → N



Gambar II.7 Gambaran *Forward Chaining*

Sumber : Andi,2009:59

II.7. Basis Data

Basis data adalah **t**empat penyimpanan data yang dipertimbangkan oleh beberapa dasar dari sebuah sistem informasi. Tujuan umum dalam merancang organisasi penyimpanan data adalah :

1. Meyakinkan pengambilan kembali data tujuan.
2. Menyediakan penyimpanan data yang efisien.
3. Ketersediaan data.
4. Mendukung pengambilan data yang efesien.

5. Menjamin integritas data. (Kendall, 2010:25)

Pertama, data harus tersedia bila pemakai ingin menggunakannya. Kedua, data harus akurat dan konsisten. Di luar syarat ini tujuan rancangan basis data termasuk penyimpanan yang efisien boleh dikatakan efisien pembaharuan dan memperoleh kembali informasi. Terakhir, penting bahwa mendapatkan informasi dengan maksud tertentu. Informasi yang diperoleh dari data yang tersimpan harus berada dalam sebuah bentuk yang berguna untuk mengatur, merencanakan, mengontrol dan membuat keputusan.

Basis data bukan hanya kumpulan *file*. Lebih dari itu, basis data adalah pusat sumber data yang caranya dipakai oleh banyak pemakai untuk berbagai aplikasi. Inti dari basis data adalah *database management system* (DBMS), yang membuktikan, pembuatan modifikasi dan pembaharuan basis data, mendapatkan kembali data dan membangkitkan laporan. Orang yang memastikan bahwa basis data memenuhi tujuannya di sebut administrator basis data. (Kendall, 2010:128)

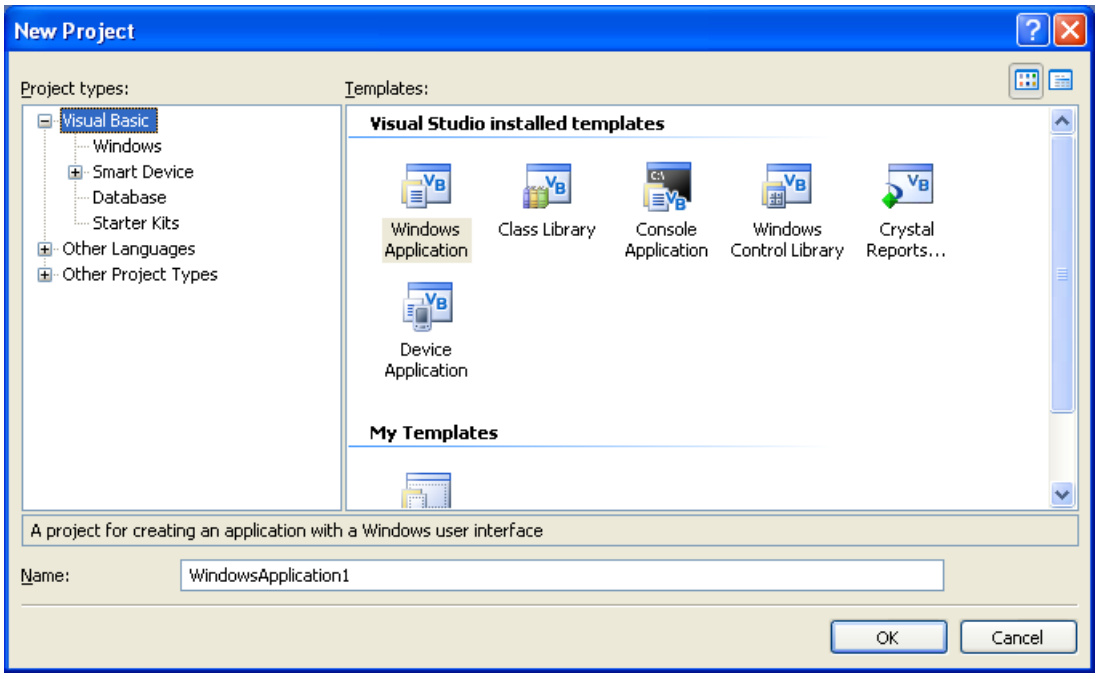
Tujuan basis data yang efektif termuat dibawah ini :

1. Memastikan bahwa data dapat dipakai diantara pemakai untuk berbagai aplikasi.
2. Memelihara data baik keakuratan maupun kekonsistenannya.
3. Memastikan bahwa semua data yang diperlukan untuk aplikasi sekarang dan yang akan datang akan disediakan dengan cepat.
4. Membolehkan basis data untuk berkembang dan kebutuhan pemakai untuk berkembang.
5. Membolehkan pemakai untuk membangun pandangan personalnya tentang data tanpa memperhatikan cara data disimpan secara fisik.(Kendall, 2010:128)

II.8. *Microsoft Visual Basic 2010*

Visual Basic.NET adalah generasi selanjutnya dari *Visual Basic*. *Visual Basic.NET* memungkinkan kita untuk membangun aplikasi *database client* atau *server* performa tinggi dan sangat cocok didampingkan dengan perangkat lunak *SQL Server* 2000.

Pemilihan *Visual Basic.NET* sebagai program pengembang sistem ini adalah karena merupakan salah satu program aplikasi yang berada di bawah *platform .NET framework*.



Gambar II.8 Contoh Tampilan *New Project*

Sumber: Kusrini, 2009:83

Dalam dialog *New Project* terdapat beberapa jenis apliksi yang akan dibuat termasuk bahasa pemrograman yang digunakan. Jenis aplikasi yang dapat dibuat adalah:

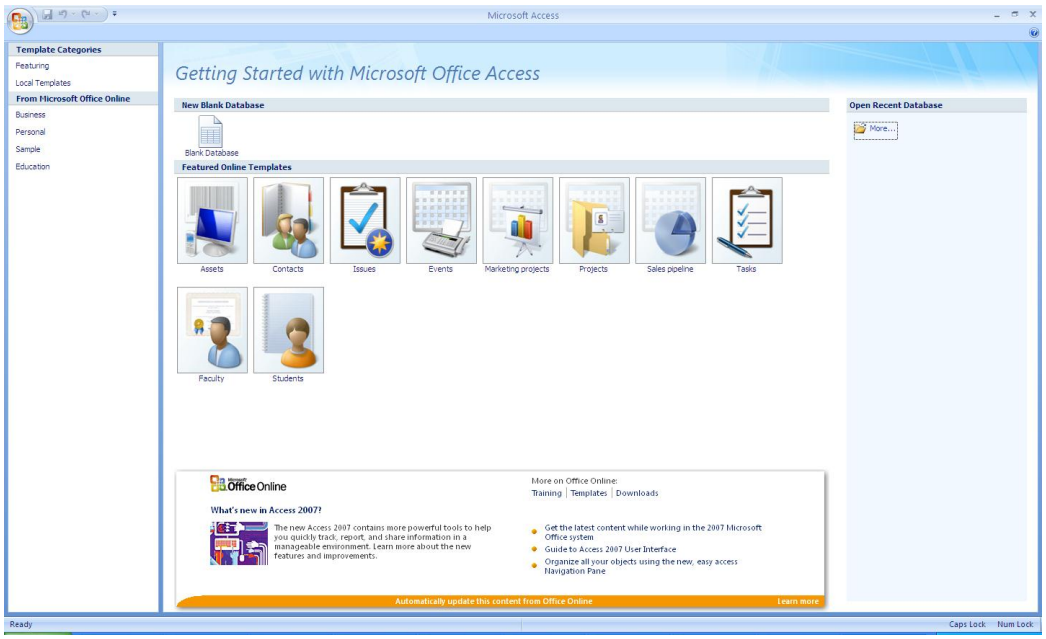
1. *Windows Application*: aplikasi yang paling umum dibuat, menggunakan *interface windows*. Biasanya, *Windows Application* merupakan *interface* aplikasi, sedangkan *logic* aplikasi terdapat di dalam *Class Library*. *Windows Application* dapat berisi *form, class, XML file,* maupun *file VB Script* dan *Jscript*.
2. *Class Library*: fondasi dasar untuk membuat komponen yang menjalankan fungsi tertentu. *Class* merupakan fondasi dasar untuk membentuk obyek dalam pemrograman berorientasi obyek. *Class Library* tidak memiliki *interface* tertentu seperti *form,* tetapi dapat diakses oleh aplikasi lain untuk menjalankan berbagai fungsi yang terdapat di dalamnya. *Class Library* dapat disamakan dengan teknologi *ActiveX DLL* (.dll) dan *ActiveX EXE* dalam pemrograman *VB6*.
3. *Windows Control Library*: tidak puas dengan built in control yang disediakan *VS.NET* . Anda dapat berkreasi membuat kontrol sendiri dan memasukkan berbagai fungsi yang Anda inginkan di dalam kontrol tersebut. Fasilitas untuk membuat kontrol tersebut adalah *Windows Control Library*. Kontrol ini sama dengan *ActiveX Control* (.ocx) dalam pemrograman *VB6*.
4. *ASP.NET Web Application*: project yang digunakan untuk membuat aplikasi web. Teknologi yang digunakan adalah *ASP.NET* yang memiliki berbagai kelebihan dibandingkan *ASP* klasik. Perubahan utamanya adalah dapat diprogram menggunakan berbagai bahasa *.NET* seperti *VB*, *C++*, *C#*, maupun *J#*. *ASP.NET* juga menyediakan berbagai kontrol yang bersifat *event driven programming* sehingga lebih menghemat waktu pembuatan aplikasi.

5. *ASP.NET Web Service*: *Web service* merupakan salah satu ide utama dalam *.NET*. Anda dapat membuat *web service* dan meletakkannya di *web server* untuk diakses berbagai aplikasi. Sebuah *web service* dapat diakses oleh aplikasi *windows*, *web*, *console*, maupun *mobile device*. *Web service* hampir sama dengan *Class Library*. Perbedaan utamanya adalah *web service* diletakkan di *web server* sehingga dapat diakses dengan lebih mudah dan tidak terbatas pada aplikasi berbasis *windows* saja.
6. *Console Application*: aplikasi dengan tampilan text mode atau DOS (*Disk Operating System*). Aplikasi jenis ini biasa digunakan sebagai *monitoring service* atau *remote application* di mana sumber daya komputer dan *bandwidth* sangat terbatas.
7. *Windows Service*: aplikasi yang berjalan sebagai di *windows*, yang di-*load* bersamaan dengan proses *start up windows*. Aplikasi ini berjalan di *background* dan biasanya tidak memiliki *interface*. Penerapan aplikasi ini misalnya untuk pembuatan *scanning antivirus*, *server FTP* dan *remote server*.
8. *Web Control Library*: hampir sama dengan *Windows Control Library* tapi digunakan untuk aplikasi *web*. (Andi, 2009:44)

II.9. *Microsoft Access 2007*

Microsoft Access adalah salah satu program aplikasi RDBMS (*Relational Database Management System*), dimana semua data yang ada disimpan dalam tabel-tabel yang terdiri dari atas lajur kolom dan baris. Dengan RDBMS, pengelolaan sebuah *database* akan mudah dilakukan walaupun jumlah datanya banyak dan kompleks. Dibandingkan dengan program aplikasi pembuatan

database lain, *Microsoft Access* sangat mudah digunakan dan fleksibel dalam pembuatan dan perancangan suatu *database*. Perancangan dan pengelolaan *database* pada *Microsoft Access* meliputi pembuatan *Table*, *Form*, *Query*, *Macro*, *Modul* dan *Pages*, yang dapat dilihat pada Gambar II.10:



Gambar II.9 Menu *Microsoft Access*

Sedangkan, tipe data yang ada dalam *Microsoft Access* adalah :

1. *Text* : untuk menerima data teks sampai 255 karakter yang terdiri dari huruf, angka dan simbol grafik.
2. *Memo* : untuk menerima data teks sampai 65,535 karakter yang terdiri dari huruf, bilangan, tanda baca serta simbol grafik. Tipe data ini tidak dapat digunakan sebagai acuan untuk pengurutan data (indeks).
3. *Number* : untuk menerima digit, tanda minus dan titik desimal. Tipe data *number* mempunyai lima pilihan ukuran bilangan dan jumlah digit tertentu.

4. *Date/Time* : untuk menerima data tanggal dan waktu, serta nilai tahun yang dimulai tahun 100 sampai dengan tahun 9999.
5. *Currency* : untuk menerima data digit, tanda minus, dan tanda titik desimal dengan tingkat ketepatan 15 digit desimal di sebelah kiri tanda titik desimal dan 4 digit di sebelah kanan tanda titik desimal.
6. *Autonumber* : untuk menampilkan nomor urut otomatis, yaitu berupa data angka mulai dari 1 dengan nilai selisih 1.
7. *Yes/No* : tipe ini untuk menerima salah satu data dari dua nilai, yaitu *Yes/No, True/False* atau *On/Off*.
8. *OLE Object* : untuk menerima data yang berupa objek grafik, *spreadsheet*, foto digital, rekaman suara atau video yang dapat diambil dari program aplikasi lain. Ukuran maksimum adalah 1 *Gigabyte*.
9. *Hyperlink* : untuk menerima data yang berupa teks yang berwarna dan bergaris bawah serta grafik dimana tipe data ini berhubungan dengan jaringan.
10. *Attachment* : untuk menerima data yang berupa *file* gambar, *spreadsheet*, dokumen, grafik dan tipe data lain.
11. *Lookup Wizard* : untuk menampilkan satu dari beberapa tipe data yang ada dalam suatu daftar. Data tersebut dapat diambil dari tabel maupun *query* yang ada. (Djoko Pramono, 2010: 79)

II.10. UML

II.10.1 Sejarah Singkat UML

UML (*Unified Modeling Language*) adalah sebuah bahasa yang berdasarkan grafik/gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan software berbasis OO (*Object-Oriented*). UML sendiri juga memberikan standar penulisan sebuah sistem *blue print*, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema database, dan komponen-komponen yang diperlukan dalam sistem *software*.

Pendekatan analisa & rancangan dengan menggunakan model OO mulai diperkenalkan sekitar pertengahan 1970 hingga akhir 1980 dikarenakan pada saat itu aplikasi *software* sudah meningkat dan mulai komplek. Jumlah yang menggunakaan metoda OO mulai diuji cobakandan diaplikasikan antara 1989 hingga 1994, seperti halnya oleh Grady Booch dari *Rational Software Co.*, dikenal dengan OOSE (*Object-Oriented Software Engineering*), serta James Rumbaugh dari *General Electric*, dikenal dengan OMT (*Object Modelling Technique*).

Kelemahan saat itu disadari oleh Booch maupun Rumbaugh adalah tidak adanya standar penggunaan model yang berbasis OO, ketika mereka bertemu ditemani rekan lainnya Ivar Jacobson dari Objectory mulai mendiskusikan untuk mengadopsi masing-masing pendekatan metoda OO untuk membuat suatu model bahasa yang uniform/seragam yang disebut UML (*Unified Modeling Language*) dan dapat digunakan oleh seluruh dunia.

Secara resmi bahasa UML dimulai pada bulan oktober 1994, ketika Rumbaugh bergabung Booch untuk membuat sebuah project pendekatan metoda

yang uniform/seragam dari masing-masing metoda mereka. Saat itu baru dikembangkan draft metoda UML version 0.8 dan diselesaikan serta di release pada bulan oktober 1995. Bersamaan dengan saat itu, Jacobson bergabung dan UML tersebut diperkaya ruang lingkupnya dengan metoda OOSE sehingga muncul release version 0.9 pada bulan Juni 1996. Hingga saat ini sejak Juni 1998 UML version 1.3 telah diperkaya dan direspons oleh OMG (Object Management Group), Anderson Consulting, Ericsson, Platinum Technology, ObjectTime Limited, dll serta di pelihara oleh OMG yang dipimpin oleh Cris Kobryn.

UML adalah standar dunia yang dibuat oleh *Object Management Group* (OMG), sebuah badan yang bertugas mengeluarkan standar-standar teknologi *object oriented* dan *software component*.
(http://wsilfi.staff.gunadarma.ac.id/Downloads/ files/1034/Pengenalan+UML.pdf)

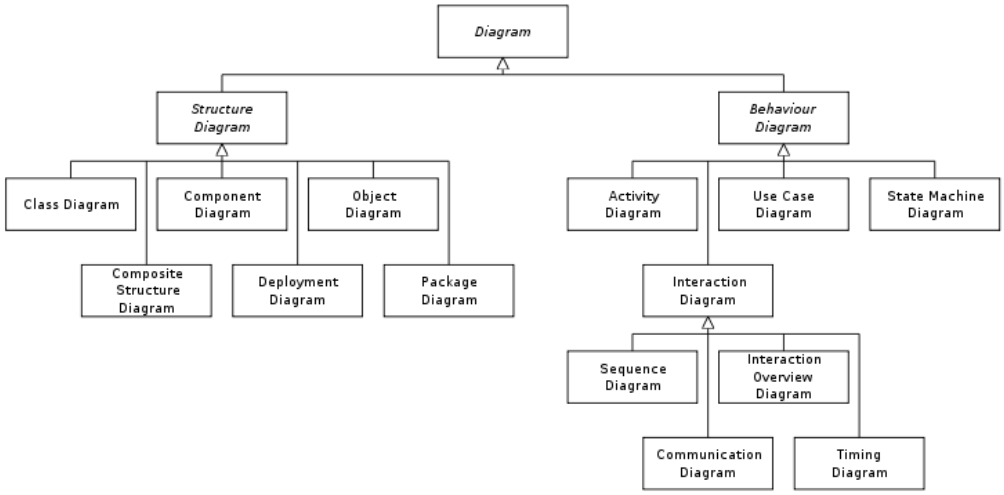
II.10.2 Pengenalan UML

Unified Modeling Language (UML) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak. UML tidak berdasarkan pada bahasa pemrograman tertentu. Standar spesifikasi UML dijadikan standar defacto oleh OMG (*Object Management Group*) pada tahun 1997. UML yang berorientasikan object mempunyai beberapa notasi standar.

Spesifikasi ini menjadi populer dan standar karena sebelum adanya UML, telah ada berbagai macam spesifikasi yang berbeda. Hal ini menyulitkan komunikasi antar pengembang perangkat lunak. Untuk itu beberapa pengembang spesifikasi yang sangat berpengaruh berkumpul untuk membuat standar baru.

UML dirintis oleh Grady Booch, James Rumbaugh pada tahun 1994 dan kemudian Ivar Jacobson. (wikipedia.org)

UML mendeskripsikan OOP (*Object Oriented Programming*) dengan beberapa diagram, seperti terlihat pada gambar 2.5 di bawah ini:



Gambar 2.5 Diagram yang digunakan pada UML

Sumber : wikipedia.org

UML sebagai sebuah bahasa yang memberikan *vocabulary* dan tatanan penulisan kata-kata dalam *'MS Word'* untuk kegunaan komunikasi. Sebuah bahasa model adalah sebuah bahasa yang mempunyai vocabulary dan konsep tatanan/aturan penulisan serta secara fisik mempresentasikan dari sebuah sistem. Seperti halnya UML adalah sebuah bahasa standard untuk pengembangan sebuah *software* yang dapat menyampaikan bagaimana membuat dan membentuk model-model, tetapi tidak menyampaikan apa dan kapan model yang seharusnya dibuat yang merupakan salah satu proses implementasi pengembangan *software*.

UML tidak hanya merupakan sebuah bahasa pemograman visual saja, namun juga dapat secara langsung dihubungkan ke berbagai bahasa pemograman, seperti JAVA, C++, Visual Basic, atau bahkan dihubungkan secara langsung ke

dalam sebuah *object-oriented database*. Begitu juga mengenai pendokumentasian dapat dilakukan seperti; *requirements*, arsitektur, *design*, *source code*, *project plan*, *tests*, dan *prototypes*.

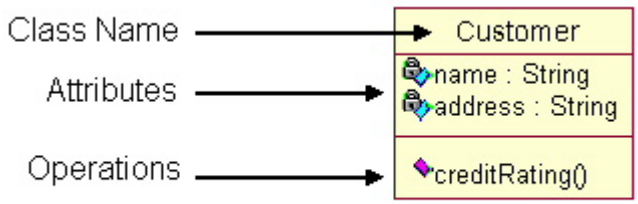
(http://wsilfi.staff.gunadarma.ac.id/Downloads/files/1034/Pengenalan+UML.pdf)

II.10.3 Class Diagram

Diagram ini mendeskripsikan struktur dari sebuah sistem dengan menunjukkan kelas-kelas dari sistem, atribut dari kelas sistem dan hubungan antara kelas sistem tersebut. *Class* adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut / properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metode / fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok, yaitu :

- Nama (dan *stereotype*)
- Atribut
- Metode atau operasi

Ketiga area pokok dari *class* dapat dirincikan sebagai berikut:



Atribut dan metode dapat memiliki salah satu sifat berikut :

- *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
- *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
- *Public*, dapat dipanggil oleh siapa saja.

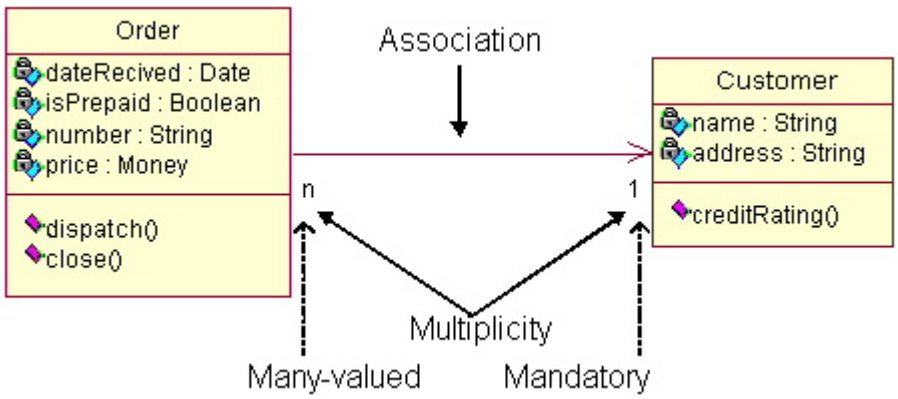
Simbol yang digunakan pada atribut dapat dirincikan sebagai berikut:

+	public
#	protected
-	private
~	package

Class diagram juga menampilkan hubungan (*relationship*) seperti:

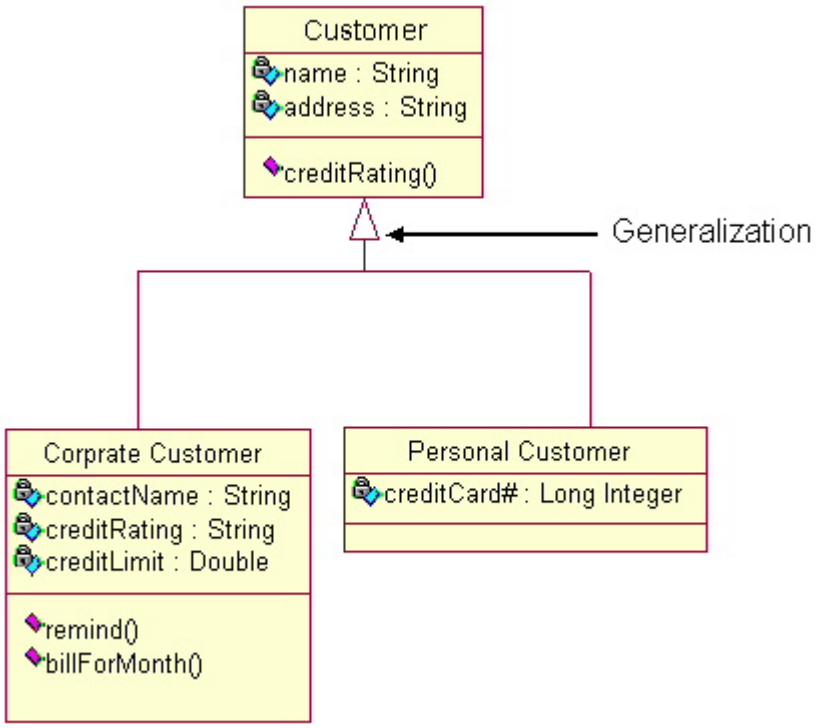
- *Associations*

  Hubungan asosiasi adalah hubungan yang paling umum pada *class diagram*. Asosiasi menunjukkan hubungan antara *instance* dari kelas. Sebagai contoh, *class order* berasosiasi dengan *class customer*.



- *Generalization*

  Generalisasi digunakan ketika dua kelas memiliki kemiripan, namun memiliki beberapa perbedaan. Sebagai contoh, dapat dilihat pada gambar berikut:



Pada contoh di atas, *class Corporate customer* dan *Personal customer* memiliki beberapa kemiripan seperti nama dan alamat, tetapi setiap *class* memiliki atribut dan operasinya sendiri. *Class customer* merupakan sebuah bentuk umum (*general form*) dari *class Corporate customer* dan *Personal customer*.

II.10.4 Activity Diagram

Diagram ini merepresentasikan *workflow* bisnis dan operasional langkah demi langkah dari komponen pada sebuah sistem. Sebuah activity diagram menunjukkan aliran kontrol secara keseluruhan.

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat

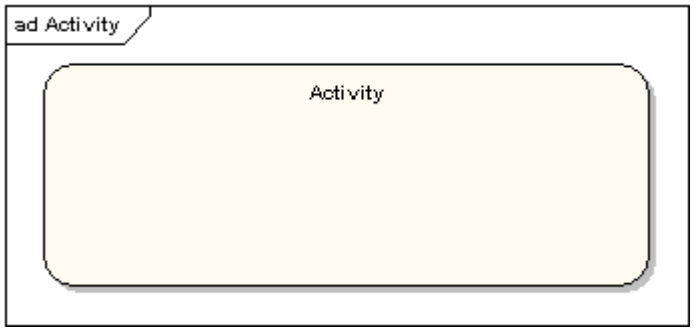
menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu, *activity diagram* tidak menggambarkan *behaviour* internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Elemen-elemen yang digunakan untuk membentuk suatu *activity diagram* adalah sebagai berikut:

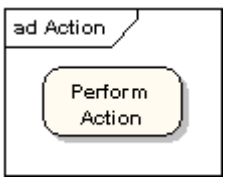
- *Activity*

  Activity adalah spesifikasi dari urutan parameter dari kelakuan. Sebuah *activity* ditunjukkan dengan menggunakan sebuah persegi panjang dengan sudut bulat dan menyertakan semua aksi, kontrol aliran dan elemen lainnya yang membentuk *activity*.



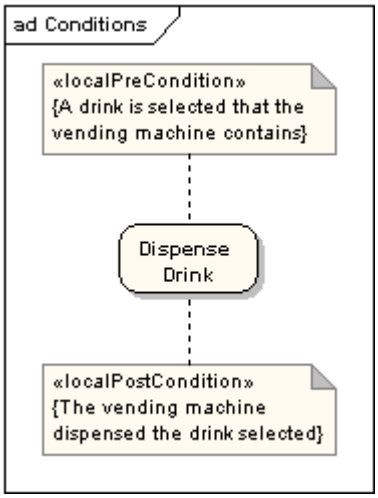
- *Action*

  Action merepresentasikan sebuah langkah tunggal diantara sebuah *activity*. Action didenotasikan dengan persegi panjang dengan sudut bulat.



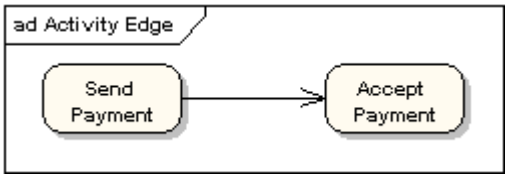
- *Action Constraint*

  Constraint dapat ditambahkan ke sebuah *action*. Diagram berikut menunjukkan sebuah *action* dengan kondisi lokal pre- dan post-.



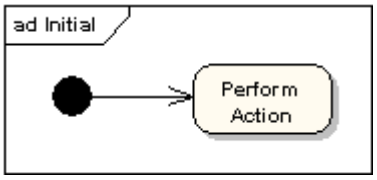
- *Control Flow*

  Control Flow menunjukkan aliran dari kontrol dari satu *action* ke berikutnya. Notasinya adalah sebuah garis dengan arah panah.



- *Initial Node*

  Sebuah *initial* atau *start node* disimbolkan dengan sebuah titik hitam besar, seperti terlihat pada gambar berikut:

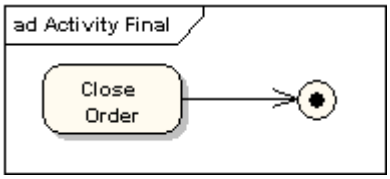


- *Final Node*

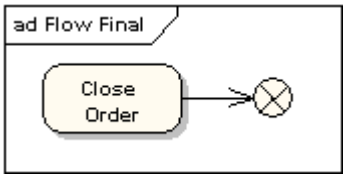
  Terdapat dua tipe dari *final node* yaitu:

  - *Activity final node* yang disimbolkan denagn sebuah lingkaran dengan

sebuah titik didalamnya.



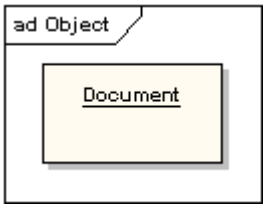
- *Flow final node* yang disimbolkan dengan sebuah lingkaran dengan sebuah tanda silang didalamnya.



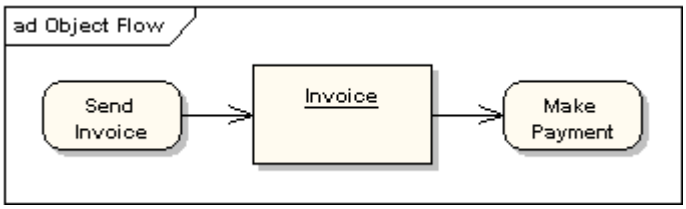
Perbedaan dari dua tipe *node* adalah *flow final node* melambangkan akhir dari sebuah aliran kontrol tunggal, sedangkan *activity final node* melambangkan akhir dari semua aliran kontrol dalam *activity*.

- *Object* dan *Object Flow*

Sebuah *object flow* adalah sebuah *path* dimana *object* atau data dapat dilewatkan. Sebuah object dilambangkan dengan sebuah persegi panjang.

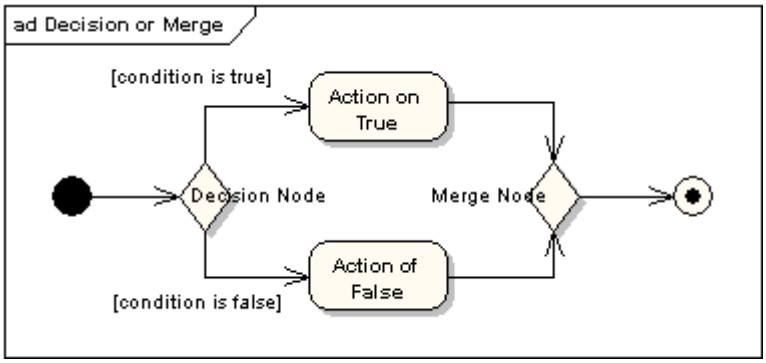


Sebuah *object flow* dilambangkan dengan sebuah *connector* dengan arah panah yang menandakan arah objek dilewatkan.



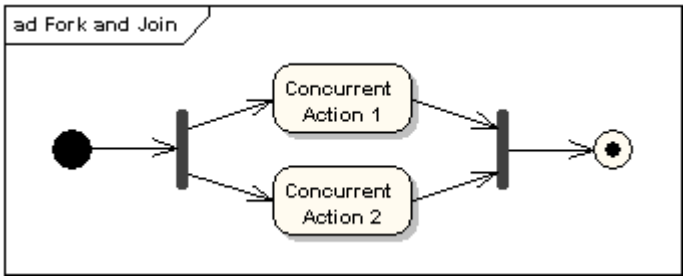
- *Decision* dan *Merge Node*

  Decision node dan *merge node* memiliki notasi yang sama yaitu bentuk diamond yang dapat diberi nama. Aliran kontrol yang datang dari sebuah *decision node* akan memiliki kondisi kunci yang mengizinkan kontrol untuk mengalir apabila kondisi kunci terpenuhi. Diagram berikut menunjukkan cara penggunaan dari *decision node* dan *merge node*.



- *Fork* dan *Join Node*

  Fork dan *join* memiliki notasi yang sama yaitu batang horizontal atau vertikal (orientasi tergantung pada apakah kontrol aliran berjalan dari kiri ke kanan atau dari atas ke bawah). Mereka mengindikasikan awal dan akhir dari urutan kontrol yang terjadi bersamaan. Diagram berikut menunjukkan contoh penggunaannya:



II.10.5 Use Case Diagram

Diagram ini menunjukkan fungsionalitas yang disediakan oleh sebuah sistem dalam bentuk *actor*, tujuan mereka yang direpresentasikan sebagai *use case* dan segala ketergantungan diantara *use case* tersebut.

Diagram *use case* adalah diagram yang menunjukkan fungsionalitas suatu sistem atau kelas dan bagaimana sistem tersebut berinteraksi dengan dunia luar dan menjelaskan sistem secara fungsional yang terlihat user. Biasanya dibuat pada awal pengembangan. *Use case* diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana".

Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-create sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

Use case diagram dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua feature yang ada pada sistem. Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-include akan dipanggil setiap kali *use case* yang meng-include dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang common. Sebuah *use case* juga dapat meng-extend *use case* lain dengan

behavior-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

Use case diagram adalah gambaran *graphical* dari beberapa atau semua *actor*, *use case*, dan interaksi diantara komponen-komponen tersebut yang memperkenalkan suatu sistem yang akan dibangun. *Use case* diagram menjelaskan manfaat suatu sistem jika dilihat menurut pandangan orang yang berada di luar sistem. Diagram ini menunjukkan fungsionalitas suatu sistem atau kelas dan bagaimana sistem tersebut berinteraksi dengan dunia luar.

Use case diagram dapat digunakan selama proses analisis untuk menangkap requirements sistem dan untuk memahami bagaimana sistem seharusnya bekerja. Selama tahap desain, *use case* diagram berperan untuk menetapkan perilaku (*behavior*) sistem saat diimplementasikan. Dalam sebuah model mungkin terdapat satu atau beberapa *use case* diagram. Kebutuhan atau *requirements* sistem adalah fungsionalitas apa yang harus disediakan oleh sistem kemudian didokumentasikan pada model *use case* yang menggambarkan fungsi sistem yang diharapkan (*use case*), dan yang mengelilinginya (*actor*), serta hubungan antara *actor* dengan *use case* (*use case* diagram) itu sendiri.

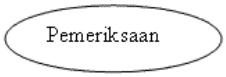
Notasi Gambar Yang Dipakai *Use case*:

- *Actor*

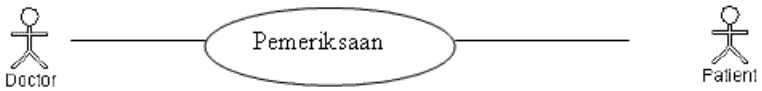
  Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

- *Case*

  Menggambarkan deskripsi yang melibatkan *actor*.



  Contoh *Case - Actor*:



- *Extend*

  Relasi yang digunakan jika *use case* yang satu mirip dengan *use case* yang lain.

- *Include*

  Relasi jika terdapat perilaku yang mirip dengan beberapa *use case*.

Cara Menemukan *Use case*:

- Pola perilaku perangkat lunak aplikasi.
- Gambaran tugas dari sebuah *actor*.
- Sistem atau "benda" yang memberikan sesuatu yang bernilai kepada *actor*.
- Apa yang dikerjakan oleh suatu perangkat lunak (bukan bagaimana cara mengerjakannya).

  (http://thoy.blogdetik.com/category/rpl/use-case-informasi-teknologi/)

II.10.6 Sequence Diagram

Diagram ini menunjukkan bagaimana objek berkomunikasi dengan satu sama lainnya dalam bentuk sebuah deretan pesan dan juga mengindikasikan waktu hidup dari objek yang berhubungan dengan pesan tersebut.

Sequence diagram menggambarkan interaksi antar objek di dalam dan di

sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

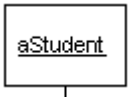
Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan.

Diagram element terbagi menjadi dua tipe yaitu:

- *Header element*, ditampilkan pada bagian *header* dari diagram.

  i. *Actor*, merepresentasikan orang atau entitas eksternal yang berinteraksi dengan sistem.

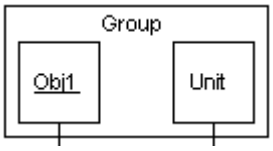
  ii. *Object*, merepresentasikan sebuah objek pada sistem atau salah satu komponennya.



  iii. *Unit*, merepresentasikan sebuah subsistem, komponen, unit atau entitas non-objek lainnya dalam sistem.

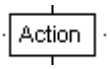
  iv. *Separator*, merepresentasikan sebuah antarmuka atau pembatas antara subsistem.

v. *Group*, menghubungkan elemen *header* ke dalam subsistem atau komponen.

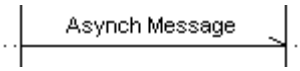


- *Body element*, ditampilkan pada badan diagram.

i. *Action*, merepresentasikan sebuah aksi yang diambil oleh sebuah elemen *header*.



ii. *Asynchronous Message*, yaitu sebuah pesan asinkron antara elemen *header*.



iii. *Block*, merepresentasikan sebuah perulangan (*loop*) atau kondisional untuk sebuah elemen header tertentu.

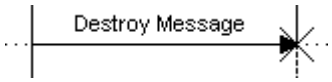
iv. *Block End*, menandai akhir dari sebuah elemen blok.

v. *Call Message*, yaitu sebuah pesan pemanggilan (prosedur) antara elemen *header*.

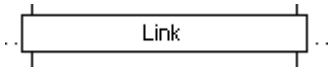
vi. *Create Message*, membuat sebuah elemen *header*.

vii. *Destroy Element*, merepresentasikan destruksi dari sebuah elemen *header*.

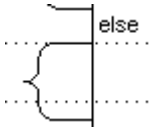
viii. *Destroy Message*, merepresentasikan destruksi dari sebuah elemen *header* sebagai akibat pemanggilan dari elemen lain.



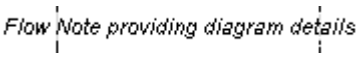
ix. *Diagram Link*, merepresentasikan sebuah seksi atau bagian dari sebuah diagram yang diperlakukan sebagai sebuah blok fungsi.



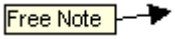
x. *Else Block*, merepresentasikan sebuah bagian blok "else" dari sebuah blok diagram.



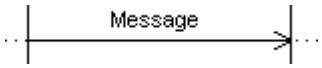
xi. *Flow Note*, yaitu catatan dokumentasi yang diformat secara otomatis ke aliran setelah elemen sebelumnya.



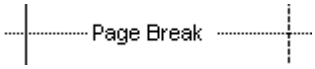
xii. *Free Note*, yaitu catatan dokumentasi yang bebas mengalir dan dapat ditempatkan di mana saja pada diagram.



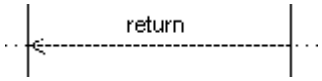
xiii. *Message*, yaitu sebuah pesan sederhana antara elemen *header*.



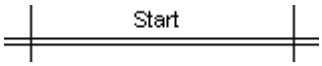
xiv. *Page Break*, yaitu sebuah pergantian halaman pada diagram.



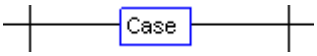
xv. *Return Message*, yaitu sebuah pengembalian pesan antara elemen *header*.



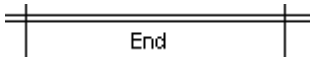
xvi. *Scenario Start*, yaitu awal dari sebuah skenario.



xvii. *Scenario Case*, yaitu awal dari sebuah alternatif atau kasus pada sebuah skenario.

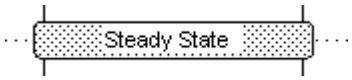


xviii. *Scenario End*, yaitu akhir dari sebuah skenario.

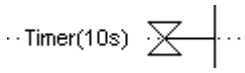


xix. *State*, yaitu sebuah perubahan *state* untuk sebuah elemen *header*.

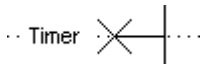
xx. *Steady State*, yaitu sebuah *state* tetap pada sistem.



xxi. *Timer Start*, yaitu awal dari *timer* untuk sebuah elemen *header* tertentu.



xxii. *Timer Stop*, yaitu penghentian dari *timer* untuk sebuah elemen *header* tertentu.



xxiii. *Timer Expiration*, akhir dari *timer* untuk sebuah elemen *header* tertentu.