BAB II

LANDASAN TEORI

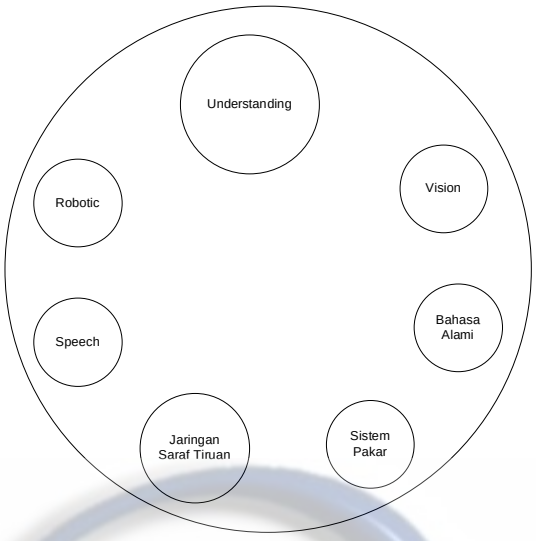
2.1 Kecerdasan Buatan

Kecerdasan buatan (Artificial Intelegence) merupakan salah satu bagian ilmu komputer yang mempelajari bagaimana cara membuat mesin (komputer) agar dapat melakukan pekerjaan seperti yang dilakukan oleh manusia. Supaya komputer dapat bertindak menyerupai manusia, maka computer harus diberi bekal pengetahuan dan mempunyai kemampuan untuk menalar. Untuk itu maka terdapat beberapa metode yang membekali komputer dengan kedua komponen tersebut agar computer menjadi sebuah mesin yang pintar.

Kecerdasan buatan berbeda dengan program konvensional. Pemrograman konvensional berbasis pada algoritma yang mendefinisikan setiap langkah dalam penyelesaian masalah. Pemrograman konvensional dapat menggunakan rumus metematika atau procedure sekuensial untuk menghasilkan solusi. Lain halnya dengan pemrograman dalam kecerdasan buatan, sebuah symbol dapat berupa kalimat, kata, atau angka yang digunakan untuk mempresentasikan obyek, proses, dan hubungannya.[4]

Obyek dapat berupa manusia, benda, ide, konsep, kegiatan, atau pernyataan dari suatu fakta. Proses digunakan untuk memanipulasi simbol untuk menghasilkan saran atau pemecahan masalah. Selain itu kecerdasan buatan dapat melakukan penalaran terhadap data yang tidak komplit.

Gambar 2.1 menunjukkan bahwa kecerdasan buatan memiliki banyak ruang lingkup atau bidang. Salah satu bidangnya adalah sistem pakar yang menggabungkan pengetahuan dan penelusuran data untuk memecahkan masalah yang secara normal memerlukan keahlian manusia.



Gambar 2.1 Ruang Lingkup Kecerdasan Buatan

2.2 Sistem Pakar

Ketika hendak membuat suatu keputusan yang komplek atau memecahkan masalah, seringkali kita meminta nasehat atau berkonsultasi dengan seorang pakar atau ahli. Seorang pakar adalah seseorang yang mempunyai pengetahuan dan pengalaman spesifik dalam suatu bidang; misalnya pakar komputer, pakar uji tak merusak, pakar politik dan lain-lain. Semakin tidak terstruktur situasinya, semakin mengkhusus (dan mahal) konsultasi yang dibutuhkan.

Sistem Pakar (Expert Sistem) adalah usaha untuk menirukan seorang pakar. Biasanya sistem pakar berupa perangkat lunak pengambil keputusan yang mampu mencapai tingkat performa yang sebanding seorang pakar dalam bidang problem yang khusus dan sempit. Ide dasarnya adalah kepakaran ditransfer dari seorang pakar (atau sumber kepakaran yang lain) ke komputer, pengetahuan yang ada disimpan dalam komputer, dan pengguna dapat berkonsultasi pada komputer itu untuk suatu nasehat, lalu komputer dapat mengambil inferensi (menyimpulkan, mendeduksi, dll.) seperti layaknya seorang pakar, kemudian menjelaskannya ke pengguna tersebut, bila perlu dengan alasan-alasannya. Sistem Pakar malahan terkadang lebih baik unjuk kerjanya dari pada seorang pakar manusia.[12]

Kepakaran (expertise) adalah pengetahuan yang ekstensif (meluas) dan spesifik yang diperoleh melalui rangkaian pelatihan, membaca, dan pengalaman. Pengetahuan membuat pakar dapat mengambil keputusan secara lebih baik dan lebih cepat dari pada non-pakar dalam memecahkan problem yang kompleks. Kepakaran mempunyai sifat berjenjang, pakar top memiliki pengetahuan lebih banyak dari pada pakar yunior.[12]

Tujuan sistem pakar adalah untuk mentransfer kepakaran dari seorang pakar ke komputer, kemudian ke orang lain (yang bukan pakar). Proses ini tercakup dalam rekayasa pengetahuan (knowledge engineering).

Suatu pengetahuan dari sistem pakar bersifat khusus untuk satu domain masalah saja. Domain masalah tersebut bersifat khusus, seperti kedokteran, keuangan, bisnis atau teknik [Sri Kusumadewi, 2003]. Sistem pakar yang baik dirancang agar dapat menyelesaikan suatu permasalahan tertentu dengan meniru kerja dari para pakar. Dengan sistem pakar, orang awam dapat menyelesaikan masalah yang cukup rumit yang sebenarnya hanya dapat diselesaikan dengan bantuan para ahli. Bagi para ahli, sistem pakar dapat membantu aktifitasnya sebagai asisten yang sangat berpengalaman.

Seorang pakar dengan sistem pakar mempunyai banyak perbedaan. Perbandingan kemampuan antara seorang pakar dengan sebuah sistem pakar dapat dilihat seperti pada tabel 2.1 berikut:

Tabel 2.1 Perbandingan seorang pakar dengan sistem pakar [5]

Pakar Manusia	Sistem Pakar
Terbatas Waktu karena manusia membutuhkan istirahat	Tidak terbatas karena dapat digunakan kapanpun juga
Tempat akses bersifat local pada suatu tempat saja dimana pakar berada	Dapat digunakan diberbagai tempat

Pakar Manusia	Sistem Pakar
Pengetahuan bersifat variable dan dapat berubah-ubah tergantung situasi	Pengetahuan bersifat konsisten
Kecepatan untuk menemukan solusi sifatnya bervariasi.	Kecepatan untuk memberikan solusi konsisten dan lebih cepat daripada manusia
Biaya yang harus dibayar untuk konsultasi biasanya sangat mahal	Biaya yang dikeluarkan lebih murah

2.3 Klasifikasi Sistem Pakar[5]

Berdasarkan kegunaannya, sistem pakar dapat diklasifikasikan menjadi enam jenis yaitu:

1. Diagnosis

Diagnosis adalah suatu tindakan atau perilaku yang menggunakan bantuan suatu alat dan sistem untuk mempelajari atau mengamati sesuatu hal yang tidak tepat, tidak teratur, sehingga menghasilkan suatu informasi dan membuat dan membuat inferensi kemungkinan penyebab timbulnya ketidak beresan.

Diagnosis sistem pakar biasanya digunakan untuk merekomendasikan obat untuk orang sakit, kerusakan mesin, kerusakan rangkaian elektronik dan sebagainya. Prisispnya adalah menemukan masalah atau kerusakan yang terjadi. Sistem pakar diagnosis merupakan jenis sistem pakar yang paling populer saat ini.

Biasanya sistem pakar diagnosis menggunakan pohon keputusan (decision tree) sebagai representasi pengetahuannya. Hal lain dari sistem pakar diagnosisi ini adalah basis pengetahuannya bertambah besar secara eksponsial dengan kompleksnya permasalahan.

2. Pengajaran

   Sistem pakar ini digunakan untuk mengajar, mulai dari murid sekolah dasar sampai mahasiswa perguruan tinggi. Kelebihan sistem pakar dapat digunakan untuk membuat diagnosa apa penyebab dari kekurangan dari seorang siswa, kemudian cara untuk memperbaikinya.

3. Interpretasi

   Sistem pakar interpretasi ini digunakan untuk menganalisa data yang tidak lengkap, tidak teratur, dan data yang kontradiktif misalnya untuk menginterpretasi citra.

4. Prediksi

   Keunggulan dari seorang pakar adalah memprediksi kedepan. Contoh yang mudah kita temui, bagaimana seorang pakar meteorologi memprediksi cuaca besok berdasrkan data-data sebelumnya. Penggunaan sistem pakar prediksi misalnya untuk peramalan cuaca, penentuan masa tanam, dan sebagainya.

5. Perencanaan

   Perencanaan sistem pakar sangat luas mulai dari perencanaan mesin-mesin sampai manajemen bisnis. Penggunaan sisem pakar jenis ini dapat menghemat biaya, waktu dan material. Contoh penggunaan anatara lain yaitu sistem konfigurasi, komputer, tata letak sirkuit.

6. Kontrol

   Sistem pakar kontrol ini digunakan untuk melakukan pengontrolan terhadap kegiatan yang membutuhkan presisi waktu yang tinggi, misalnya pada industri-industri berteknologi tinggi.

2.4 Manfaat, Keterbatasan dan Alasan Pengembangan Sistem Pakar

2.4.1 Manfaat Sistem Pakar

Mengapa Sistem Pakar menjadi sangat populer? Hal ini disebabkan oleh sangat banyaknya kemampuan dan manfaat yang diberikan oleh Sistem Pakar, di antaranya:

a. Meningkatkan output dan produktivitas, karena sistem pakar dapat bekerja lebih cepat dari manusia.
b. Meningkatkan kwalitas, dengan memberi nasehat yang konsisten dan mengurangi kesalahan.
c. Mampu menangkap kepakaran yang sangat terbatas.
d. Dapat beroperasi di lingkungan yang berbahaya.
e. Handal. Sistem Pakar tidak pernah menjadi bosan dan kelelahan atau sakit. Sistem Pakar juga secara konsisten melihat semua detil dan tidak akan melewatkan informasi yang relevan dan solusi yang potensial.
f. Mampu bekerja dengan informasi yang tidak lengkap atau tidak pasti. Berbeda dengan sistem komputer konvensional, Sistem Pakar dapat bekerja dengan inofrmasi yang tidak lengkap. Pengguna dapat merespon dengan: "ya" atau "tidak" pada satu atau lebih pertanyaan selama konsultasi, dan Sistem pakar tetap akan memberikan jawabannya.

2.4.2 Keterbatasan Sistem Pakar

Metodologi Sistem Pakar yang ada tidak selalu mudah, sederhana dan efektif. Berikut adalah keterbatasan yang menghambat perkembangan Sistem Pakar:

1. Pengetahuan yang hendak diambil tidak selalu tersedia.
2. Kepakaran sangat sulit diekstrak dari manusia.
3. Pendekatan oleh setiap pakar untuk suatu situasi atau problem bisa berbeda-beda, meskipun sama-sama benar.
4. Adalah sangat sulit bagi seorang pakar untuk mengabstraksi atau menjelaskan langkah mereka dalam menangani masalah.
5. Sistem Pakar bekerja baik untuk suatu bidang yang sempit.

6. Banyak pakar yang tidak mempunyai jalan untuk mencek apakah kesimpulan mereka benar dan masuk akal.
7. Pengembangan sistem pakar seringkali membutuhkan perekayasa pengetahuan (knowledge engineer) yang langka dan mahal.
8. Kurangnya rasa percaya pengguna menghalangi pemakaian sistem pakar.

2.4.3 Alasan Pengembangan Sistem Pakar[3]

Sistem pakar sendiri dikembangkan lebih lanjut dengan alasan :

1. Dapat menyediakan kepakaran setiap waktu dan di berbagai lokasi.
2. Secara otomatis mengerjakan tugas-tugas rutin yang membutuhkan seorang pakar.
3. Seorang pakar akan pensiun atau pergi.
4. Seorang pakar adalah mahal.
5. Kepakaran dibutuhkan juga pada lingkungan yang tidak bersahabat.

2.5 Struktur Sistem Pakar

Struktur sistem pakar disusun oleh dua bagian utama, yaitu lingkungan pengembangan (development environment) dan lingkungan konsultasi (consultation environment) (Turban, 1995). Lingkungan pengembangan sistem pakar digunakan untuk memasukkan pengetahuan pakar ke dalam lingkungan sistem pakar, sedangkan lingkungan konsultasi digunakan oleh pengguna yang bukan pakar guna memperoleh pengetahuan pakar.

2.5.1 Antarmuka Pengguna (User Interface)

Sistem Pakar mengatur komunikasi antara pengguna dan komputer. Komunikasi ini paling baik berupa bahasa alami, biasanya disajikan dalam bentuk tanya-jawab dan kadang ditampilkan dalam bentuk gambar/grafik. Antarmuka yang lebih canggih dilengkapi dengan percakapan (voice communication).

2.5.2 Akuisisi Pengetahuan

Akuisis 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 di bidangnya, laporan dan literatur. Sumber pengetahuan tersebut dijadikan dokumentasi untuk dipelajari, diolah dan diorganisasikan secara terstruktur menjadi basis pengetahuan.

Setelah proses akuisisi pengetahuan selesai dilakukan, maka pengetahuan tersebut harus direpresentasikan menjadi basis pengetahuan dan basis aturan yang selanjutnya dikumpulkan, dikodekan, diorganisasikan, dan digambarkan dalam bentuk rancangan lain menjadi bentuk yang sistematis.

2.5.3 Basis Pengetahuan

Basis pengetahuan mengandung pengetahuan untuk pemahaman, formulasi, dan penyelesaian masalah. Komponen sistem pakar ini disusun atas dua elemen dasar, yaitu fakta dan aturan. Fakta merupakan informasi tentang obyek dalam area permasalahan tertentu, sedangkan aturan merupakan informasi tentang cara bagaimana memperoleh fakta baru dari fakta yang telah diketahui.

Dalam studi kasus pada sistem berbasis pengetahuan terdapat beberapa karakteristik yang dibangun untuk membantu kita di dalam membentuk serangkaian prinsip-prinsip arsitekturnya.

2.5.4 Mesin Inferensi

Mekanisme 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, mekanisme inferensi menguji aturan satu demi satu sampai kondisi aturan itu benar.

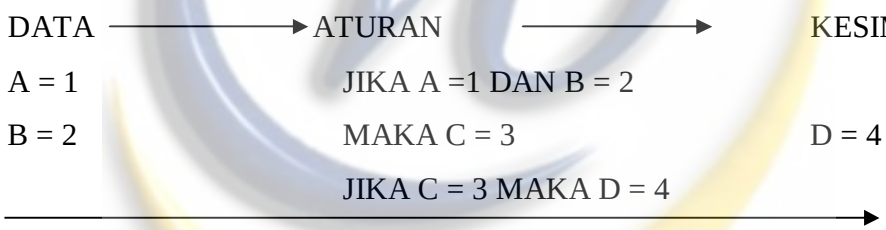
Secara umum ada dua teknik utama yang digunakan dalam mekanisme inferensi untuk mendapatkan solusi bagi permasalahan yang dihadapi sistem pakar,

yaitu runut maju (forward chaining) dan runut balik (backward chaining). Berikut adalah pendekatan untuk mengontrol inferensi dan metode penelusuran yang akan digunakan.

1. Runut Maju (forward chaining)

Dalam penalaran maju, aturan-aturan di uji satu demi satu dalam urutan tertentu. Urutan itu mungkin berupa pemasukan aturan kedalam basis aturan atau juga urutan lain yang ditentukan oleh pemakai. Saat setiap aturan diuji, sistem pakar akan mengevaluasi apakah kondisinya benar atau salah.

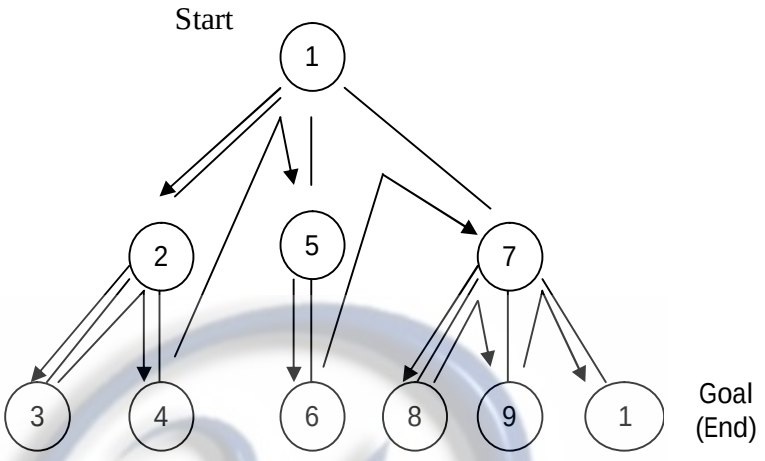
Jika kondisinya benar, maka aturan itu disimpan kemudiaan aturan berikutnya diuji. Sebaliknya kondisinya salah, aturan itu tidak disimpan kemudian aturan berikutnya di uji. Proses ini akan berulang (iterative) sampai seluruh basis aturan teruji dengan berbagai kondisi. Runut maju memulai proses pencarian dengan data sehingga strategi ini disebut juga data-driven. Gambar berikut menunjukkan bagaimana cara kerja metode inferensi runut maju.



Gambar 2.2 Runut Maju (forward chaining)

2. Depth-first search

Melakukan penelusuran kaidah secara mendalam dari simpul akar bergerak menurun ke tingkat yang dalam berurutan seperti pada gambar 2.3 di bawah ini.



Gambar 2.3 Diagram Alir Teknik Penelusuran Depth First Search[1]

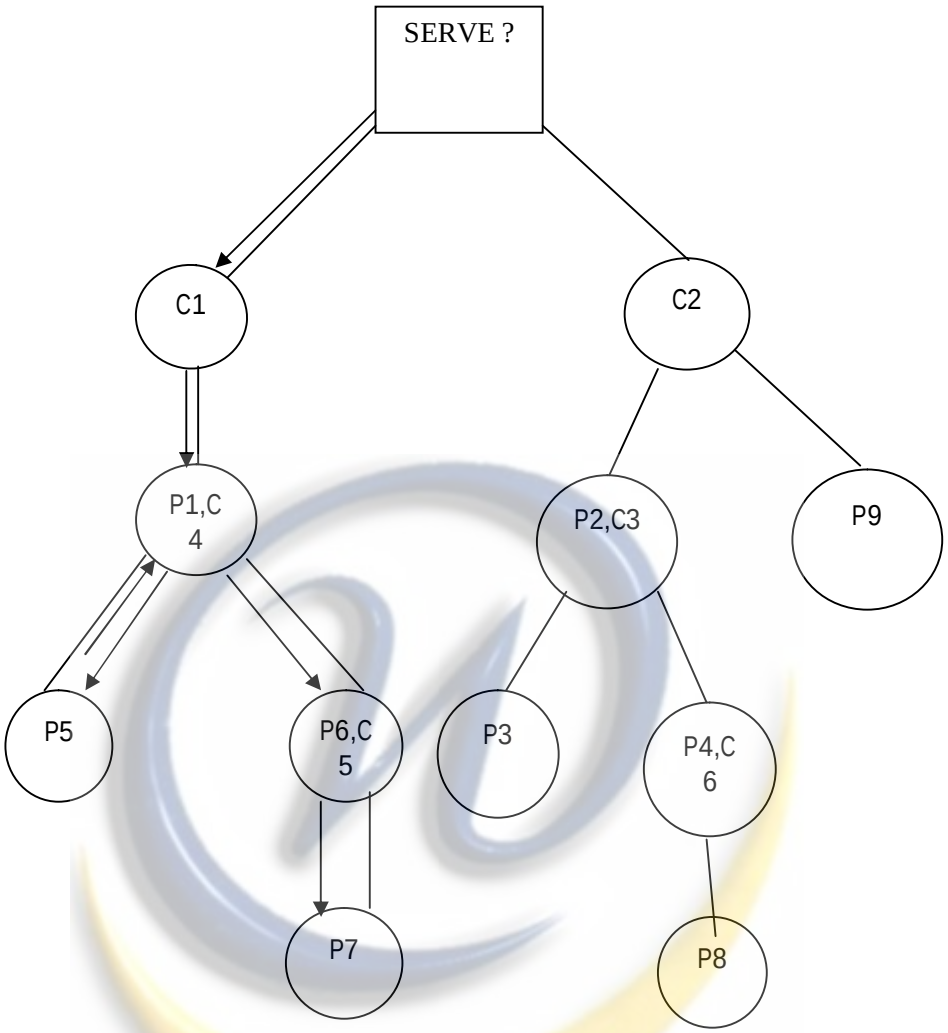
Contoh kasus dan penggunaan dari teknik penelusuran Depth-First Search. Permasalahan dalam contoh kasus ini yaitu untuk memilih jenis anggur yang tepat untuk disediakan bersama makanan, apakah anggur merah atau putih.

RULES	IF	AND	THEN
Aturan 1	Pelanggan membeli daging – P1		Pelanggan harus disediakan anggur merah – C1
Aturan 2	Pelanggan membeli ikan – P2		Pelanggan harus disediakan anggur putih – C2
Aturan 3	Toko mempunyai ikan – P3	Pelanggan mempunyai uang untuk membeli ikan – P4	Pelanggan membeli ikan – C3
Aturan 4	Toko mempunyai daging – P5	Pelanggan mempunyai uang	Pelanggan membeli daging – C4

RULES	IF	AND	THEN
		untuk membeli daging – P6	
Aturan 5	Pelanggan mempunyai uang lebih atau sama dengan $10 – P7		Pelanggan mempunyai uang untuk membeli daging – C5
Aturan 6	Pelanggan mempunyai uang antara $5 dan $10 – P8		Pelanggan mempunyai uang untuk membeli ikan
Aturan 7	Pelanggan benar-benar menyukai anggur putih – P9		Pelanggan harus disediakan anggur putih – C 2

Tabel 2.2 Contoh Kasus Pemilihan Anggur

Pada sistem forward chaining, sistem mulai melakukan pencarian satu keadaan atau node pada jaringan, nama node tersebut adalah P1, P2, P3, P5, P7, P8, atau P9. Kemudian melakukan pencarian ke cabang-cabang secara menyeluruh ke node lain untuk melihat apakah hal tersebut dapat memposisikan pada node tujuan.



Gambar 2.4 Depth First Search dari aturan-aturan pemilihan anggur[4]

Pada contoh diatas, pencarian dimulai dengan mencoba untuk menentukan anggur mana yang akan disediakan dengan makanan; tujuan dari sistem. Pertama-tama memilih node C1, kemudian melakukan pencarian ke bawah melewati node-node yang terhubung untuk mencari informasi pendukung. Jika informasi tersebut ditemukan maka pencarian dihentikan. Sebaliknya, pencarian dilanjutkan dari kiri ke kanan.

Seperti pada gambar 2.4, masalah berhasil diselesaikan setelah menguji kelima node. Sebuah rekomendasi dari anggur merah (C1) telah dibuat karena toko mempunyai daging (P5) dan uang lebih besar atau sama dengan $10 (P7). Sejak

sistem telah dapat menemukan sebuah solusi di C1, node C2 dan semua node dibawahnya (keturunannya) tidak dilakukan pencarian.

2.5.5 Workplace

Workplace merupakan area dari sekumpulan memori kerja (working memory). Workplace digunakan untuk merekam hasil-hasil antara dan kesimpulan yang dicapai. Ada 3 tipe keputusan yang dapat direkam, yaitu :

1. Rencana : Bagaimana menghadapi masalah.
2. Agenda : Aksi-aksi yang potensial yang sedang menunggu untuk dieksekusi.
3. Solusi : Calon aksi yang akan dibangkitkan.

2.5.6 Perbaikan Pengetahuan

Pakar memiliki kemampuan untuk menganalisis dan meningkatkan kinerjanya serta kemampuan untuk belajar dan kinerjanya. Kemampuan tersebut adalah penting dalam pembelajaran terkomputerisasi, sehingga program akan mampu menganalisis penyebab kesuksesan dan kegagalan ulang dialaminya.

2.5.7 Representasi Pengetahuan

Representasi pengetahuan adalah suatu teknik untuk merepresentasikan basis pengetahuan yang diperoleh ke dalam suatu skema/diagram tertentu sehingga dapat diketahui relasi/keterhubungan antara suatu data dengan data yang lain[1]. Teknik ini membantu knowledge engineer dalam memahami struktur pengetahuan yang akan dibuat sistem pakarnya.

Ada beberapa cara merepresentasikan data menjadi basis pengetahuan, seperti yang dikemukakan oleh Barr dan Feigenbaum pada tahun 1981, yaitu data dalam bentuk kalkulus predikat, bingkai, kaidah produksi, representasi logika, jaringan semantik. Semua bentuk representasi data tersebut bertujuan untuk menyederhanakan data sehingga mudah dimengerti dan mengefektifkan proses pengembangan program. Dalam melakukan representasi pengetahuan penulis menggunakan kaidah produksi.

Metode kaidah produksi biasanya dituliskan dalam bentuk jika-maka (if-then). Kaidah ini dapat dikatakan sebagai hubungan implikasi dua bagian, yaitu bagian premise (jika) dan bagian konklusi (maka). Apabila bagian premise dipenuhi maka bagian konklusi juga akan bernilai benar. Sebuah kaidah terdiri dari klausa-klausa. Sebuah klausa mirip sebuah kalimat dengan subyek, kata kerja dan obyek yang menyatakan suatu fakta. Ada sebuah klausa premise dan sebuah klausa konklusi pada setiap kaidah. Suatu kaidah juga terdiri atas beberapa premise dan lebih dari satu konklusi. Antara premise dan konklusi dapat dihubungkan dengan “atau” atau “dan”.

2.6 Pemrograman Microsoft Visual Basic 6.0

Microsoft Visual Basic 6.0 merupakan suatu bahasa pemrograman yang memberikan berbagai fasilitas pembuatan aplikasi visual. Keunggulan bahasa pemrograman ini terletak pada produktivitas, kualitas, pengembangan perangkat lunak, kecepatan kompilasi, pola desain yang menarik serta diperkuat dengan pemrogramannya yang terstruktur. Keunggulan lain dari Microsoft Visual Basic 6.0 ini adalah dapat digunakan untuk merancang program aplikasi yang memiliki tampilan seperti program aplikasi lain yang berbasis Windows. Khusus untuk pemrograman database, Microsoft Visual Basic 6.0 menyediakan fasilitas objek yang kuat dan lengkap yang memudahkan programmer dalam membuat program. Bentuk database yang dimiliki Microsoft Visual Basic 6.0 adalah bentuk database Paradox, dBase, MS.Access, ODBC, Fox Pro, Excel dan lain-lain.

Lingkungan pengembangan terpadu atau Integrated Development Environment (IDE) dalam program Microsoft Visual Basic 6.0 terbagi menjadi empat bagian yaitu:

a. Form Perantara Pemakai

Dalam Visual Basic, form adalah sebuah jendela yang dapat diatur untuk membuat perantara pemakai dari program. Dalam program Step up, form adalah jendela yang dilihat pada saat bekerja. Sebuah form dapat berisi menu, tombol, kotak

daftar, baris penggulung, dan item lainnya yang ada pada program bebasis window pada umumnya.

b. Toolbox

Untuk menambahkan elemen-elemen perantara pemakai program ke sebuah form memakai alat Bantu tool, atau kontrol, dalam toolbox, yang umumnya terletak di sebelah kiri layar. Setelah kontrol ditambahkan ke sebuah form, kontrol ini menjadi sebuah objek, atau elemen perantara pemakai yang dapat di program, dalam program tersebut. Toolbox berisi kontrol-kontrol yang dapat kita pakai untuk menambah artwork, label, tombol-tombol, kotak daftar, baris penggulung, kisi-kisi, menu, dan bentuk-bentuk geometric ke perantara pemakai. Elemen-elemen ini akan terlihat ketika program dijalankan. Toolbox juga berisi objek-objek yang melaksanakan operasi khusus ”dibelakang layar” objek ini tidak akan terlihat meskipun program sudah dijalankan.

c. Jendela Properties

Memungkinkan untuk mengubah karakteristik, atau pengaturan property, dari elemen-elemen perantara pemakai pada sebuah form. Pengaturan properties adalah kualitas salah satu objek dalam perantara pemakai. Jendela properties berisi sebuah kotak daftar objek, yang mendaftarkan semua elemen perantarapemakai (objek) yang ada pada form. Jendela properties juga mendaftarkan peraturan property yang dapat diubah untuk setiap objek.

d. Jendela Project

Program Visual Basic tersusun dari beberapafile yang dirangkai bersama, atau di kompilasi, jika program telah lengkap. Jendela project berfungsi untuk membantu perpindahan antar komponen saat mengerjakan program dalam lingkungan pemprograman.

Kode-kode program yang akan diletakkan pada objek yang akan menggunakan kode tersebut pada setiap kejadian yang kita inginkan. Kode tersebut akan dijalankan setiap terjadi sesuatu atas objek tersebut selama Running Time.

Setiap kejadian yang terjadi atas objek tersebut selanjutnya disebut Event Handler. Setiap aplikasi Windows selalu digerakkan oleh pesan (message). Pesan ini dikirimkan oleh Windows ke aplikasi dan aplikasi memberikan respon karena pesan yang diterimanya.

Cara ini merupakan teknik yang dilakukan oleh Windows untuk implementasi aplikasi-aplikasi yang berada dalam lingkungannya, terutama untuk manajemen sistem supaya beberapa program dapat dijalankan pada saat yang bersamaan (multitasking).

2.7 Perancangan System dengan UML (Unified Modeling Language)

2.7.1 Use Case Diagram

Diagram yang bersifat statis, diagram ini memperlihatkan himpunan use case dan actor (suatu jenis khusus dari kelas) diagram ini sangat penting untuk mengorganisasikan dan memodelkan prilaku dari suatu sistem yang dibutuhkan serta diharapkan pengguna. Adapun simbol-simbol yang sering digunakan dalam Use Case Diagram adalah :

Tabel 2.3 Simbol-simbol pada Use Case Diagram

No.	Simbol	Keterangan
1.		Simbol Actor, menggambarkan aktor pada diagram.
2.		Simbol UseCase, menggambarkan UseCase pada diagram
3.		Simbol Unidirectional Association, menggambarkan relasi antar aktor dan use case

2.7.2 Class Dalam Model Analisis

Elemen model yang terdapat dalam model analisis disebut kelas analisis (analysis class). Kelas analisis adalah kelas ber-stereotype “Boundary”, “Control”, atau “Entity” yang menggambarkan sebuah konsep awal mengenai “benda” dalam sistem aplikasi yang memiliki tanggung jawab dan perilaku. Kelas analisi akhirnya berkembang menjadi kelas didalam model desain. Adapun simbol - simbol yang sering digunakan dalam Class dalam model analisis adalah :

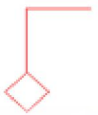
Tabel 2.4 Simbol-simbol pada Class Model Analisis

No.	Simbol	Keterangan
1.		Simbol Boundary, menggambarkan batasan kelas pada diagram. Dimana kelas yang memodelkan interaksi antara satu atau lebih actor dengan sistem
2.		Simbol Control, menggambarkan unsure kendali pada diagram.
3.		Entity menggambarkan kelas entitas pada diagram.

2.7.3 Class Diagram

Diagram yang bersifat statis, diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi, serta relasi-relasi. Diagram ini umum dijumpai pada pemodelan berorientasi objek, meskipun bersifat statis sering pula diagram kelas ini memuat kelas-kelas aktif. Adapun simbol-simbol yang sering digunakan dalam Class Diagram adalah :

Tabel 2.5 Simbol-simbol pada Class Diagram

No.	Simbol	Keterangan
1.		Simbol class, menggambarkan aktor pada diagram.
2.		Simbol Agregation, menggambarkan relasi agregasi
3.		Simbol Association, menggambarkan relasi asosiasi.

2.7.4 Sequence Diagram

Diagram yang bersifat dinamis, diagram urutan adalah diagram interaksi yang menekankan pada pengiriman pesan dalam suatu waktu tertentu. Adapun simbol-simbol yang sering digunakan dalam Sequence Diagram adalah :

Tabel 2.6 Simbol-simbol pada Sequence Diagram

No.	Simbol	Keterangan
1.		Simbol Actor, menggambarkan aktor pada diagram.
2.		Simbol Boundary, menggambarkan batasan kelas pada diagram.
3.		Simbol Control, menggambarkan unsure kendali pada diagram.

No.	Simbol	Keterangan
4.		Entity menggambarkan kelas entitas pada diagram.
5.		Object Message, menggambarkan pesan antar dua objek.
6.		Message to Self, menggambarkan pesan yang menuju dirinya sendiri.

2.7.5 Collaboration Diagram

Diagram yang bersifat dinamis, diagram kolaborasi adalah diagram interaksi yang menekankan organisasi struktural dari objek-objek yang menerima serta mengirim pesan. Adapun simbol-simbol yang sering digunakan dalam Collaboration Diagram adalah :

Tabel 2.7 Simbol-simbol pada Collaboration Diagram

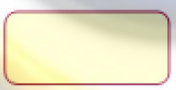
No.	Simbol	Keterangan
1.		Simbol Actor, menggambarkan aktor pada diagram.
2.		Simbol Boundary, menggambarkan batasan kelas pada diagram.
3.		Simbol Control, menggambarkan unsure kendali pada diagram.
4.		Entity menggambarkan kelas entitas pada diagram.

No.	Simbol	Keterangan
5.		Link to Self, menggambarkan bahwa suatu objek memanggail operasinya sendiri
6.		Object Link, menggambarkan lintasan komunikasi antar dua objek.
7.		Link Message, menggambarkan pesan antar dua objek, atau dari suatu objek ke dirinya sendiri.
8.		Reverse Link Message, menggambarkan pesan dalam arah berlawanan antar dua objek. atau dari suatu objek ke dirinya sendiri.

2.7.6 Activity Diagram

Diagram yang bersifat dinamis, diagram aktivitas adalah tipe khusus dalam diagram state yang memperlihatkan aliran dari sesuatu aktifitas ke aktfitas lainnya dalam suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi-fungsi dalam suatu sistem dan memberi tekanan pada aliran kendali antar objek. Adapun simbol-simbol yang sering digunakan dalam Activity Diagram adalah :

Tabel 2.8 Simbol-simbol pada Activity Diagram

No.	Simbol	Keterangan
1.		Simbol Start state, menggambarkan aliran kerja berawal.
2.		Simbol End state, menggambarkan aliran kerja berakhir.
3.		Simbol Decision, menggambarkan Titik keputusan pada aliran kerja.
4.		State Transition, menggambarkan transisi dari suatu aktivitas ke aktivitas yang lain.
5.		State, menggambarkan state untuk suatu objek.

2.7.7 Statechart Diagram

Diagram yang bersifat dinamis, diagram ini memperlihatkan state, transisi event, serta aktifitas. Diagram ini penting terutama untuk memperlihatkan sifat dinamis dari antar muka (interface), kelas, kolaborasi, dan terutama penting pada pemodelan sistem-sistem yang reaktif. Adapun simbol-simbol yang sering digunakan dalam Statechart Diagram adalah :

Tabel 2.9 Simbol-simbol pada Statechart Diagram

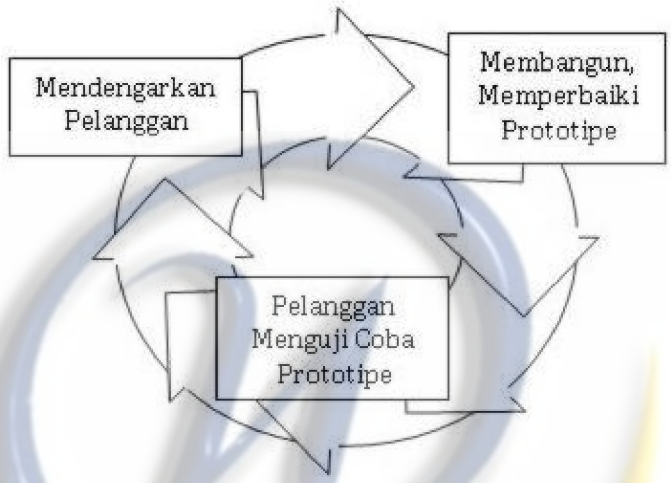
No.	Simbol	Keterangan
1.		Simbol Start state, menggambarkan state awal. Pada diagram.
2.		Simbol End state, menggambarkan state berakhir pada diagram.
3.		Simbol Transition to Self, menggambarkan transisi yang mengarah pada state tunggal.
4.		State Transition, menggambarkan transisi pada diagram.

2.8 Metode Rekayasa Perangkat Lunak dengan Model Prototype (Prototyping Model)[14]

Dahulu, rancangan fisik merupakan proses yang menggunakan kertas dan pinsil. Seorang analis mengambarkan tata letak atau struktur dari output, input, basis data, dan aliran hubungan dan prosedur. Ini merupakan proses yang memakan waktu yang memiliki kemungkinan terjadinya kesalahan. Biasanya hasil dari rancangan kertas ini adalah tidak lengkap dan tidak akurat. Sekarang, banyak analis dan perancang memilih Prototyping, sebuah pendekatan berbasis rekayasa (engineering) untuk merancang. Pendekatan Prototyping adalah proses iterative yang melibatkan hubunan kerja yang dekat antara perancang dan pengguna.

Pressman (2001) menyatakan bahwa seringkali seorang pelanggan mendefinisikan serangkaian sasaran umum bagi perangkat lunak, tetapi tidak

mengidentifikasi kebutuhan input, pemrosesan, ataupun output detail. Pada kasus yang lain, pengembang mungkin tidak memiliki kepastian terhadap efisiensi algoritme, kemapuan penyesuaian dari sistem operasi, atau bentuk-bentuk yang harus dilakukan oleh interaksi manusia dan mesin. Dalam situasi seperti ini salah satu model yang cocok digunakan adalah model prototype (Prototyping paradigm). Model Prototype dapat dilihat pada gambar dibawah ini.



Gambar 2.5 Metode RPL dengan Model Prototype

Pendekatan Prototyping melewati tiga proses, yaitu pengumpulan kebutuhan, perancangan, dan evaluasi Prototype. Proses-proses tersebut dapat dijelaskan sebagai berikut:

1. Pengumpulan kebutuhan: developer dan klien bertemu dan menentukan tujuan umum, kebutuhan yang diketahui dan gambaran bagian-bagian yang akan dibutuhkan berikutnya;
2. Perancangan: perancangan dilakukan cepat dan rancangan mewakili semua aspek software yang diketahui, dan rancangan ini menjadi dasar pembuatanprototype;
3. Evaluasi Prototype: klien mengevaluasi prototype yang dibuat dan digunakan untuk memperjelas kebutuhan software.

Perulangan ketiga proses ini terus berlangsung hingga semua kebutuhan terpenuhi. prototype-prototype dibuat untuk memuaskan kebutuhan klien dan untuk

memahami kebutuhan klien lebih baik. Prototype yang dibuat dapat dimanfaatkan kembali untuk membangun software lebih cepat, namun tidak semua prototype bisa dimanfaatkan. Sekalipun prototype memudahkan komunikasi antar developer dan klien, membuat klien mendapat gambaran awal dari Prototype. Pendekatan ini memiliki beberapa keuntungan :

1. Pemodelan membutuhkan partisipasi aktif dari end-user. Hal ini akan meningkatkan sikap dan dukungan pengguna untuk pengerjaan proyek. Sikap moral pengguna akan meningkat karena system berhubungan nyata dengan mereka.
2. Perubahan dan iterasi merupakan konsekuensi alami dari pengembangan system-sehingga end user memiliki keinginan untuk merubah pola pikirnya.Prototyping lebih baik menempatkan situasi alamiah ini karena mengasumsikan perubahan model melalui iterasi kedalam system yang dibutuhkan.
3. Prototyping mematahkan folosofi "end user tidak mengetahui secara detail apa yang dibutuhkan sampai mereka melihat implementasinya"
4. Prototyping adalah model aktif, tidak pasif, sehingga end user dapat melihat, merasakan, dan mengalaminya.
5. Kesalahan yang terjadi dalam prototyping dapat dideteksi lebih dini
6. Prototyping dapat meningkatkan kreatifitas karena membolehkan adanyafeedback dari end user. Hal ini akan memberikan solusi yang lebih baik.
7. Prototyping mempercepat beberapa fase hidup dari programmer.

McLeod dan Schell (2001) mengemukakan bahwa alasan-alasan pemakai maupun spesialis informasi menyukai model prototype adalah:

1. Komunikasi antara analis sistem dan pemakai membaik;
2. Analis dapat bekerja dengan lebih baik dalam menemukan kebutuhan pemakai;
3. Pemakai berperan lebih aktif dalam pengembangan sistem;

4. Spesialis informasi dan pemakai menghabiskan lebih sedikit waktu dan usaha dalam mengembangkan sistem;
5. Implementasi menjadi lebih mudah karena pemakai mengetahui sistem yang diharapkan.

Tetapi, terdapat beberapa kelemahan dari prototyping, kelemahan tersebut antara lain :

1. Prototyping memungkinkan terjadinya pengembalian terhadap kode, implementasi, dan perbaikan siklus hidup yang dugunakan untuk mendominasi sistem informasi.
2. Prototyping tidak menolak kebutuhan dari fase analisis sistem. Prototype hanya dapat memecahkan masalah yang salah dan memberi kesempatan sebagai sistem pengembangan konvensional.
3. Perancangan issu numerik tidak dialamaykan oleh prototyping. Isu tersebut dapat dilupakan jika pengguna tidak berhati-hati.
4. Prototyping dapat mengurangi kreatifitas perancangan.

Prototyping terkadang dapat memberikan performansi yang lambat, membantu mendapatkan kebutuhan detil lebih baik namun demikian Prototype juga menimbulkan masalah:

1. Dalam membuat prototype banyak hal yang diabaikan seperti efisiensi, kualitas, kemudahan dipelihara/dikembangkan, dan kecocokan dengan lingkungan yang sebenarnya. Jika klien merasa cocok dengan prototype yang disajikan dan berkeras terhadap produk tersebut, maka developer harus kerja keras untuk mewujudkan produk tersebut menjadi lebih baik, sesuai kualitas yang seharusnya.
2. Developer biasanya melakukan kompromi dalam beberapa hal karena harus membuat prototype dalam waktu singkat. Mungkin sistem operasi yang tidak sesuai, bahasa pemrograman yang berbeda, atau algoritma yang lebih sederhana.

3. Agar model ini bisa berjalan dengan baik, perlu disepakati bersama oleh klien dan developer bahwa prototype yang dibangun merupakan alat untuk mendefinisikan kebutuhan software.