BAB II

LANDASAN TEORI

2.1 Kecerdasan Buatan (*Artificial Intellegence*)

Kecerdasan Buatan (bahasa Inggris: *Artificial Intelligence* atau *AI*)menurut wikipedia didefinisikan sebagai kecerdasan yang ditunjukkan oleh suatu *entitas* buatan. Sistem seperti ini umumnya dianggap komputer. Kecerdasan diciptakan dan dimasukkan ke dalam suatu mesin (komputer) agar dapat melakukan pekerjaan seperti yang dapat dilakukan manusia. Beberapa macam bidang yang menggunakan kecerdasan buatan antara lain sistem pakar, permainan komputer (*games*), *logika fuzzy*, jaringan syaraf tiruan dan robotika.

Banyak hal yang kelihatannya sulit untuk kecerdasan manusia, tetapi untuk Informatika relatif tidak bermasalah. Seperti contoh: mentransformasikan persamaan, menyelesaikan persamaan integral, membuat permainan catur atau Backgammon. Di sisi lain, hal yang bagi manusia kelihatannya menuntut sedikit kecerdasan, sampai sekarang masih sulit untuk direalisasikan dalam Informatika. Seperti contoh: Pengenalan Obyek/Muka, bermain Sepakbola.

Walaupun AI memiliki konotasi fiksi ilmiah yang kuat, AI membentuk cabang yang sangat penting pada ilmu komputer, berhubungan dengan perilaku, pembelajaran dan adaptasi yang cerdas dalam sebuah mesin. Penelitian dalam AI menyangkut pembuatan mesin untuk mengotomatisasikan tugas-tugas yang membutuhkan perilaku cerdas. Termasuk contohnya adalah pengendalian, perencanaan dan penjadwalan, kemampuan untuk menjawab diagnosa dan

pertanyaan pelanggan, serta pengenalan tulisan tangan, suara dan wajah. Hal-hal seperti itu telah menjadi disiplin ilmu tersendiri, yang memusatkan perhatian pada penyediaan solusi masalah kehidupan yang nyata. Sistem AI sekarang ini sering digunakan dalam bidang ekonomi, obat-obatan, teknik dan militer, seperti yang telah dibangun dalam beberapa aplikasi perangkat lunak komputer rumah dan video game.

2.1.1 Dasar Pemikiran Kecerdasan Buatan

Secara garis besar, AI terbagi ke dalam dua faham pemikiran yaitu AI Konvensional dan Kecerdasan Komputasional (CI, *Computational Intelligence*). AI konvensional kebanyakan melibatkan metoda-metoda yang sekarang diklasifiksikan sebagai pembelajaran mesin, yang ditandai dengan formalisme dan analisis statistik. Dikenal juga sebagai AI simbolis, AI logis, AI murni dan AI cara lama (GOFAI, *Good Old Fashioned Artificial Intelligence*). Metoda-metodanya menurut wikipedia meliputi:

1) Sistem pakar: menerapkan kapabilitas pertimbangan untuk mencapai kesimpulan. Sebuah sistem pakar dapat memproses sejumlah besar informasi yang diketahui dan menyediakan kesimpulan-kesimpulan berdasarkan pada informasi-informasi tersebut.
2) Petimbangan berdasar kasus
3) Jaringan Bayesian
4) AI berdasarkan tingkah laku: metoda modular pada pembentukan sistem AI secara manual

Kecerdasan komputasional melibatkan pengembangan atau pembelajaran iteratif (misalnya penalaan parameter seperti dalam sistem koneksionis. Pembelajaran ini berdasarkan pada data empiris dan diasosiasikan dengan AI non-simbolis, AI yang tak teratur dan perhitungan lunak. Metoda-metoda pokok AI menurut wikipedia [10] meliputi:

1) Jaringan Syaraf: sistem dengan kemampuan pengenalan pola yang sangat kuat
2) Sistem Fuzzy: teknik-teknik untuk pertimbangan di bawah ketidakpastian, telah digunakan secara meluas dalam industri modern dan sistem kendali produk konsumen.
3) Komputasi Evolusioner: menerapkan konsep-konsep yang terinspirasi secara biologis seperti populasi, mutasi dan "survival of the fittest" untuk menghasilkan pemecahan masalah yang lebih baik.

Metoda-metoda ini terutama dibagi menjadi algoritma evolusioner (misalnya algoritma genetik) dan kecerdasan berkelompok (misalnya algoritma semut). Dengan sistem cerdas hibrid, percobaan-percobaan dibuat untuk menggabungkan kedua kelompok ini. Aturan inferensi pakar dapat dibangkitkan melalui jaringan syaraf atau aturan produksi dari pembelajaran statistik seperti dalam ACT-R. Sebuah pendekatan baru yang menjanjikan disebutkan bahwa penguatan kecerdasan mencoba untuk mencapai kecerdasan buatan dalam proses pengembangan evolusioner sebagai efek samping dari penguatan kecerdasan manusia melalui teknologi.

2.1.2 Sejarah *Kecerdesan Buatan*

Pada awal abad 17, René Descartes mengemukakan bahwa tubuh hewan bukanlah apa-apa melainkan hanya mesin-mesin yang rumit. Blaise Pascal menciptakan mesin penghitung digital mekanis pertama pada 1642. Pada 19, Charles Babbage dan Ada Lovelace bekerja pada mesin penghitung mekanis yang dapat diprogram.

Bertrand Russell dan Alfred North Whitehead menerbitkan *Principia Mathematica,* yang merombak logika formal. Warren McCulloch dan Walter Pitts menerbitkan "Kalkulus Logis Gagasan yang tetap ada dalam Aktivitas " pada 1943 yang meletakkan pondasi untuk jaringan syaraf.

Tahun 1950-an adalah periode usaha aktif dalam AI. Program AI pertama yang bekerja ditulis pada 1951 untuk menjalankan mesin Ferranti Mark I di University of Manchester (UK): sebuah program permainan naskah yang ditulis oleh Christopher Strachey dan program permainan catur yang ditulis oleh Dietrich Prinz. John McCarthy membuat istilah "kecerdasan buatan" pada konferensi pertama yang disediakan untuk pokok persoalan ini, pada 1956. Dia juga menemukan bahasa pemrograman Lisp. Alan Turing memperkenalkan "Turing test" sebagai sebuah cara untuk mengoperasionalkan test perilaku cerdas. Joseph Weizenbaum membangun ELIZA, sebuah chatterbot yang menerapkan psikoterapi Rogerian.

Selama tahun 1960-an dan 1970-an, Joel Moses mendemonstrasikan kekuatan pertimbangan simbolis untuk mengintegrasikan masalah di dalam program *Macsyma,* program berbasis pengetahuan yang sukses pertama kali dalam bidang matematika. Marvin Minsky dan Seymour Papert menerbitkan

Perceptrons, yang mendemostrasikan batas jaringan syaraf sederhana dan Alain Colmerauer mengembangkan bahasa komputer *Prolog*. Ted Shortliffe mendemonstrasikan kekuatan sistem berbasis aturan untuk representasi pengetahuan dan inferensi dalam diagnosa dan terapi medis yang kadangkala disebut sebagai sistem pakar pertama. Hans Moravec mengembangkan kendaraan terkendali komputer pertama untuk mengatasi jalan berintang yang kusut secara mandiri.

Pada tahun 1980-an, jaringan syaraf digunakan secara meluas dengan algoritma perambatan balik, pertama kali diterangkan oleh Paul John Werbos pada 1974. Tahun 1990-an ditandai perolehan besar dalam berbagai bidang AI dan demonstrasi berbagai macam aplikasi. Lebih khusus *Deep Blue*, sebuah komputer permainan catur, mengalahkan Garry Kasparov dalam sebuah pertandingan 6 game yang terkenal pada tahun 1997. DARPA menyatakan bahwa biaya yang disimpan melalui penerapan metode AI untuk unit penjadwalan dalam Perang Teluk pertama telah mengganti seluruh investasi dalam penelitian AI sejak tahun 1950 pada pemerintah AS.

2.2 Sistem Pakar

Sistem pakar adalah suatu program komputer yang mengandung pengetahuan dari satu atau lebih pakar manusia mengenai suatu bidang spesifik. Jenis program ini pertama kali dikembangkan oleh periset kecerdasan buatan pada dasawarsa 1960-an dan 1970-an dan diterapkan secara komersial selama 1980-an. Bentuk umum sistem pakar adalah suatu program yang dibuat berdasarkan suatu set aturan yang menganalisis informasi mengenai suatu kelas masalah spesifik

serta analisis matematis dari masalah tersebut. Tergantung dari desainnya, sistem pakar juga mampu merekomendasikan suatu rangkaian tindakan pengguna untuk dapat menerapkan koreksi. Sistem ini memanfaatkan kapabilitas penalaran untuk mencapai suatu simpulan.

2.2.1 Pengertian

Sistem pakar adalah sistem berbasis komputer yang menggunakan pengetahuan, fakta, dan teknik penalaran dalam memecahkan masalah masalah yang biasanya hanya dapat dipecahkan oleh seorang pakar dalam bidang tersebut. Adapun beberapa definisi tentang Sistem Pakar (Expert System) lainnya, adalah sebagai berikut:

a) Menurut Kus [4], sistem pakar didefinisikan sebagai berikut: “Sistem pakar (Expert system) adalah sistem yang berusaha mengadopsi pengetahuan manusia ke komputer, agar komputer dapat menyelesaikan masalah seperti yang biasa dilakukan oleh para ahli”.

b) Menurut Sus [13], sistem pakar didefinisikan sebagai berikut: “Sistem pakar adalah program komputer yang beroperasi menyerupai pengetahuan seorang pakar dalam bentuk heuristik”.

c) Menurut Sup [15], sistem pakar didefinisikan sebagai berikut: “Sistem pakar adalah program kecerdasan buatan yang menggabungkan pangkalan pengetahuan (Knowledge Base) dengan sistem inferensi”.

d) Menurut Umar [14], sistem pakar didefinisikan sebagai berikut: “Sistem pakar biasa disebut juga dengan “Knowledge-based System” yaitu suatu

aplikasi komputer yang ditujukan untuk pengambilan keputusan atau pemecahan persoalan dalam bidang yang spesifik".

Pada dasarnya sistem pakar diterapkan untuk mendukung aktivitas pemecahan masalah. Beberapa pemecahan yang dimaksud antara lain: pembuatan keputusan (decision making), pemaduan pengetahuan (knowledge fusing), pembuatan desain (designing), perencanaan (planning), prakiraan (forecasting), pengaturan (regulating), pengendalian (controlling), diagnosis (diagnosing), perumusan (prescribing), penjelasan (explaining), pemberian nasihat (advising), dan pelatihan (tutoring). Selain itu sistem pakar juga dapat berfungsi sebagai asisten yang pandai dari seorang pakar.

Sistem pakar dibuat pada wilayah pengetahuan tertentu untuk suatukepakaran tertentu yang mendekati kemampuan manusia di salah satu bidang. Sistem pakar mencoba mencari solusi yang memuaskan sebagaimana yang dilakukan seorang pakar. Selain itu, sistem pakar juga dapat memberikan penjelasan terhadap langkah yang diambil dan memberikan alasan atas saran atau kesimpulan yang ditemukannya.

Biasanya sistem pakar hanya digunakan untuk memecahkan masalah yang sulit untuk dipecahkan dengan pemrograman biasa, mengingat biaya yang diperlukan untuk membuat sistem pakar jauh lebih besar dari pembuatan sistem biasa.

2.2.2 Ciri-ciri sistem pakar

Ciri-ciri sistem pakar adalah sebagai berikut:

1. Terbatas pada domain keahlian tertentu.
2. Dapat memberikan penalaran untuk data data yang tidak pasti.
3. Dapat mengemukan rangkaian alasan-alasan yang diberikannya dengan cara yang dapat dipahami.
4. Berdasarkan pada kaidah/Rule tertentu.
5. Dirancang untuk dapat dikembangkan secara bertahap.
6. Keluaranya bersifat anjuran.

Menurut Turban [11] Komponen sistem pakar terbagi menjadi empat bagian, yaitu:

1. Knowledge Base (Basis Pengetahuan)

   Knowledge Base merupakan inti dari program sistem pakar karena basis pengetahuan itu merupakan presentasi pengetahuan atau knowledge representation. Basis pengetahuan adalah sebuah basis data yang menyimpan aturan-aturan tentang suatu domain knowledge/pengetahuan tertentu. Basis pengetahuan ini terdiri dari kumpulan objek beserta aturan dan atributnya (sifat atau cirinya). Contoh : If hewan merupakan sayap dan bertelur then hewan jenis burung.

2. Working Memory (Basis Data atau Memori Kerja)

   Working memory adalah bagian yang mengandung semua fakta-fakta baik fakta awal pada saat sistem beroperasi maupun fakta-fakta pada saat pengambilan kesimpulan sedang dilaksanakan selama sistem pakar beroperasi basis data berada di dalam memori kerja.

3. Inference Engine (Mesin Inferensia)

   Inference Engine adalah bagian yang menyediakan mekanisme fungsi berfikir dan pola-pola penalaran sistem yang digunakan oleh seorang pakar.

   1. Mekanisme ini akan menganalisa masalah tertentu dan selanjutnya akan mencari jawabanatau kesimpulan yang terbaik.
   2. Mesin ini akan dimulai pelacakannya dengan mencocokan kaidah-kaidah dalam basis pengetahuan dengan fakta-fakta yang ada dalam basis data.

   Dua teknik Inference, yaitu:

   a. *Backward Chaining* (Pelacakan kebelakang)

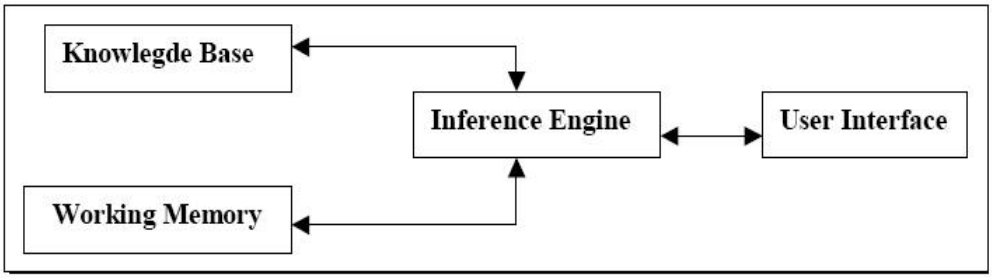
   Melalui penalaranya dari sekumpulan hipotesis menuju fakta-fakta yang mendukung tersebut, jadi proses pelacakan berjalan mundur dimulai dengan menentukan kesimpulan yang akan dicari baru kemudian fakta-fakta pembangun kesimpulan atau *a Goal Driven.*

   b.*Forward Chaining* (Pelacakan ke depan)

   Forward Chaining merupakan kebalikan dari *Backward Chaining* yaitu mulai dari kumpulan data menuju kesimpulan. Suatu kasus kesimpulannya dibangun berdasarkan fakta-fakta yang telah diketahui atau data driven.

4. User Interface (Antarmuka Pemakai)

   Antarmuka pemakai adalah bagian penghubung antara program sistem pakar dengan pemakai. Pada bagian memungkinkan pengguna untuk memasukkan instruksi dan informasi ke dalam sistem pakar serta menerima penjelasan dan kesimpulan (gambar 2.1).



Gambar 2.1 User Interface

2.2.3 Keuntungan Sistem Pakar

Secara garis besar, banyak manfaat yang dapat diambil dengan adanya sistem pakar, antara lain:

1) Membuat seorang awam dapat bekerja seperti layaknya seorang pakar.
2) Memiliki kemampuan untuk memecahkan masalah yang kompleks.
3) Bisa melakukan proses secara berulang secara otomatis.
4) Meningkatkan kualitas.
5) Membuat peralatan yang kompleks lebih mudah dioperasikan, karena sistem pakar dapat melatih pekerja yang tidak berpengalaman.
6) Memiliki reliabilitas.
7) Meningkatkan output dan produktivitas.
8) Menghemat waktu dalam pengambilan keputusan.
9) Meningkatkan kapabilitas dalam penyelesaian masalah.

10) Sebagai media pelengkap dalam pelatihan.
11) Memiliki kemampuan untuk bekerja dengan informasi yang tidak lengkap dan mengandung ketidakpastian.
12) Sistem pakar tidak dapat lelah atau bosan.
13) Mampu mengambil dan melestarikan keahlian para pakar (terutama yang termasuk keahlian langka).
14) Mampu beroperasi dalam lingkungan yang berbahaya.
15) Memiliki kemampuan untuk mengakses pengetahuan.
16) Meningkatkan kapabilitas sistem komputer.
17) Sistem pakar menyediakan nasihat yang konsisten dan dapat mengurangi tingkat kesalahan.
18) Meningkatkan kemampuan problem solving, karena mengambil sumber pengetahuan dari banyak pakar.
19) Meniadakan kebutuhan perangkat yang mahal.
20) Fleksibel.

2.2.4 Kekurangan Sistem Pakar

Disamping memiliki beberapa keuntungan, sistem pakar juga memiliki beberapa kelemahan, antara lain:

1) Biaya yang diperlukan untuk membuat dan memeliharanya sangat mahal.
2) Sulit dikembangkan. Hal ini tentu saja erat kaitannya dengan ketersediaan pakar di bidangnya.
3) Sistem pakar tidak 100% bernilai benar.

2.2.5 Kategori Sistem Pakar

Berikut Merupakan Tabel kategori dari sistem pakar :

Tabel 2.1 Kategori Sistem Pakar

Kategori	Masalah yang diselesaikan.
Diagnosis	Menduga kegagalan sistem dari observasi Contoh: Mendiagnosa kerusakan mesin mobil.
Interprestasi	Menduga gambaran situasi dari observasi.
Prediksi	Menduga akibat yang mungkin terjadi dari situasi tertentu. Contoh: Prediksi Cuaca besok berdasarkan data-data sebelumnya.
Perencanaan	Mengembangkan rencana-rencana untuk mencapai tujuan tertentu.
Monitoring	Membandingkan observasi terhadap rencana hasil.
Debugging	Memberikan obat bagi kegagalan fungsi.
Instruksi	Mendiagnosa, mununjukkan unjuk kerja.
Kontrol	Mengimplementasikan, memprediksi, mengontrol kegiatan yang membutuhkan presisi yang tinggi.

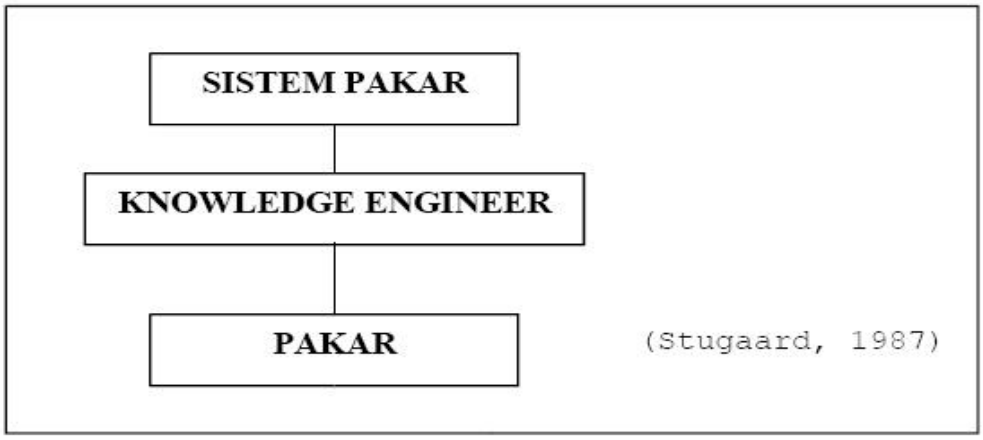
2.2.6 Sistem Kerja Pakar

Menurut Staugard (1987) sistem kerja pakar terbagi dalam tiga modul yaitu:

a. Modul Penerimaan Pengetahuan

Untuk mendapatkan pengetahuan sistem pakar dilakukan proses penerimaan pengetahuan. Proses ini dilakukan melalui interaksi dengan pakar penerimaan pengetahuan dilakukan dengan bantuan *Knowledge*

Engineer (KE), yaitu seorang spesialis sistem yang menterjemahkan pengetahuan yang dimiliki seorang pakar menjadi pengetahuan yang akan tersimpan dalam basis pengetahuan pada sebuah sistem pakar (gambar 2.2).



Gambar 2.2 Knowledge Engineer

b. Modul Konsultasi

Sistem pakar pada modul konsultasi apabila sistem memberikan konsultasi berupa jawaban atas permasalahan yang diajukan oleh pemakai pada modul ini pemakai yang awam berinteraksi dengan sistem dengan memasukkan data dan jawaban-jawaban pertanyaan sistem. Data yang dimasukkan oleh pemakai ditempatkan dalam database sistem dan kemudian diakses oleh pembangkit inference untuk mendapatkan kesimpulan.

c. Modul Penjelasan

Modul Penjelasan adalah menjelaskan proses pengambilan keputusan yang dilakukan oleh sistem

2.2.7 Cara Representasi

Cara representasi dalam sistem pakar (Turban,1992) terbagi dalam tiga teknik, yaitu:

a. *Production Rule*

Production Rule adalah model ide dasar dari sistem yang memopresentasikan pengetahuan dengan bentuk pasangan kondisi aksi (Jika-Maka).

b. *Semantic Network / Decision Tree* (Pohon keputusan)

Semantic Network / Decision Tree adalah gambaran grafis dari pengetahuan yang terdiri node atau symbol dan hubungan atau link yang memperlihatkan hubungan hirarkis antar objek.

c. *Frame*

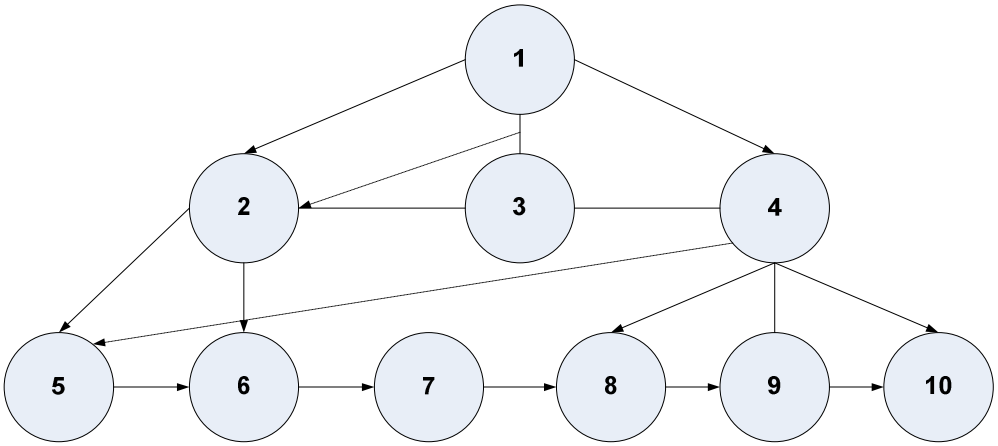
Frame adalah struktur data yang berisi semua pengetahuan tentang objek tertentu.

2.2.8 Metoda Pencarian dan Pelacakan

Metoda pencarian yang digunakan dalam sistem pakar ini menggunakan *forward chainning* dan teknik pelacakan pertama mendalam (*depth first search*). Namun selain dua teknik tersebut terdapat pula teknik pelacakan melebar pertama (*breadth first search*). Namun dalam pembuatan sistem pakar ini teknik yang digunakan adalah pelacakan pertama mendalam (*depth first search*).

a. Pelacakan melebar pertama (*breadth first search*)

Pelacakan (*breadth first search*) menguji semua noda dalam pohon pelacakan. Noda yang ada pada setiap tingkat seluruhnya diuji sebelum pindah ke tingkat berikutnya. Pelacakan *breadth first search* sederhana digambarkan dalam gambar



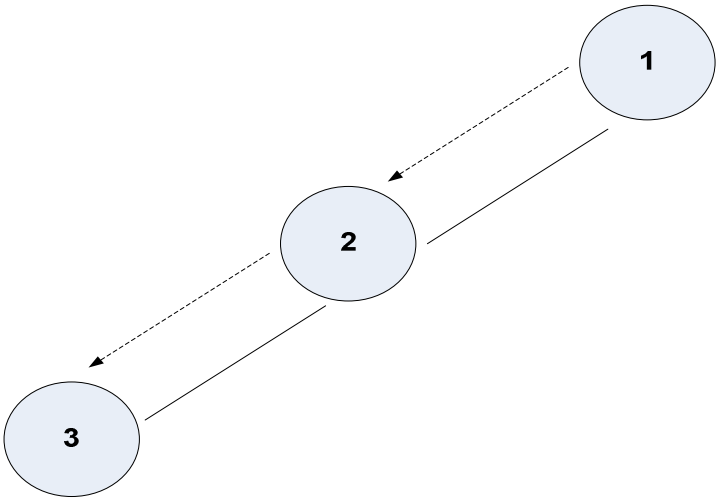
Gambar 2.3 Pelacakan melebar pertama (*breadth first search*)

Sumber Kus [4]

Pelacakan ini berangkat dari noda awal dan pertama menurun, kemudian menelusur dari kiri ke kanan.

b. Pelacakan pertama mendalam (*depth first search*)

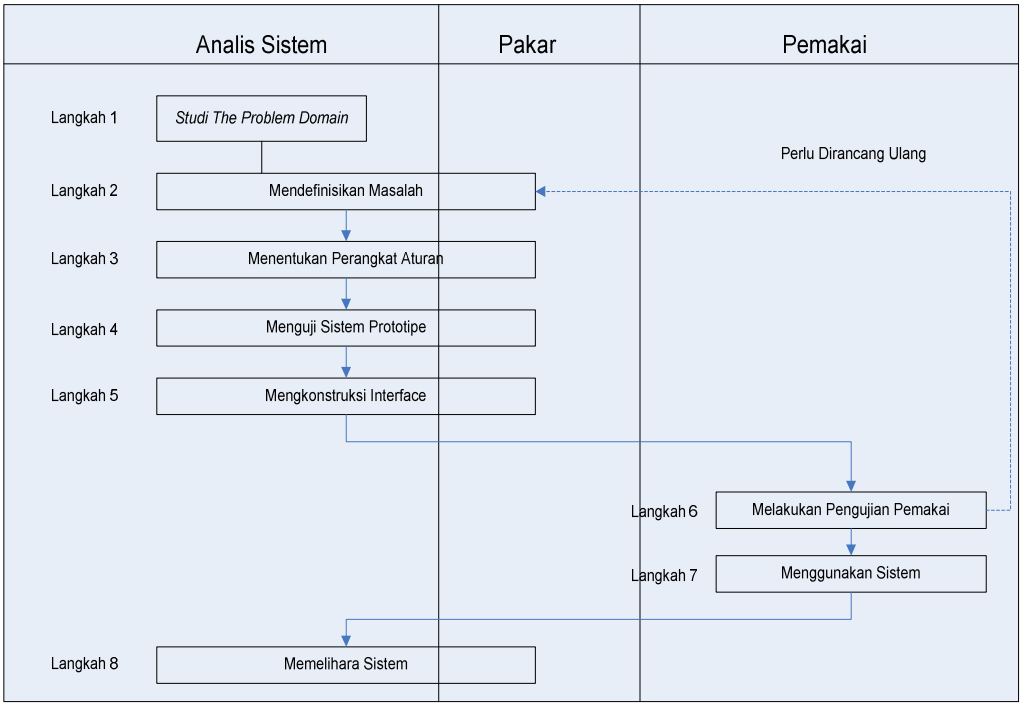
Proses pencarian akan dilakukan pada semua level anaknya sebelum dilakukan pencarian ke node – node yang selevel, pencarian dimulai dari node akar ke level yang lebih tinggi. Proses ini diulangi terus sehingga diketemukan solusi. Pelacakan depth first search digambarkan sebagai berikut



Gambar 2.4 Pelacakan pertama mendalam (*depth first search*).

Sumber Kus [4]

2.2.9 Tahap – Tahap perancangan sistem pakar



Gambar 2.5 Tahap – tahap perancangan sistem pakar
Sumber Jr [2]

Gambar tersebut menjelaskan bagaimana analisis sistem bekerja sama dengan pakar untuk memulai proses pengembangan dan mengembangkan prototipe sistem pakar, dan kemudian memperoleh partisipasi pemakai.

Memulai proses pengembangan tugas analisa sistem dalam langkah 1 adalah mempelajari *problem domain* dan menentukan apakah pekerjaan pengembangan harus dilakukan. Jika ya analis menjadi cukup tahu tentang problem domain untuk melanjutkan proyek.

Mengembangkan prototipe sistem pakar. Dalam langkah 2 sampai 5 analis bekerjasama dengan pakar. Dalam beberapa kasus manajer yang mengidentifikasi kebutuhan sistem bertindak sebagai pakar, atau pakar dapat merupakan manajer lain dalam perusahaan atau seseorang dari luar seperti konsultan, atau satu atau beberapa manajer perusahaan lain. Bahkan mungkin perlu meminta bantuan beberapa pakar untuk mendapatkan pengetahuan yang dibutuhkan. Analis dan pakar mengikuti pendekatan sistem dengan pertama – tama mendefinisikan masalah pada langkah 2. pada langkah 3 mereka menemukan perangkat aturan dan menyatukan aturan-aturan tersebut ke dalam prototipe.

Analis dan pakar menguji prototipe pada langkah 4, dan jika pengujian tidak berjalan baik prototipe diperbaiki, dimulai lagi dengan langkah 2. jika pengujian berjalan seperti yang diharapkan, analis dan pakar menambahkan *user interface* pada langkah 5

Partisipasi pemakai. Selanjutnya pemakai perlu melakukan pengujian pada prototipe. Sistem analis dan pakar memberikan pelatihan yang diperlukan bagi pemakai dan pada langkah 6 pengujian dilakukan. Jika pemakai tidak puas dengan hasilnya, analis dan pakar melakukan pendekatan baru dimulai dengan langkah 2

jika pengujian berjalan baik, pemakai dapat menggunakan sistem pakar pada langkah 7.

Pemeliharaan sistem pakar. Sistem pakar harus dipelihara seperti subsistem CBIS yang lain, dan hal ini dapat dilakukan pada langkah 8.

2.3 Konsep Basis Data

2.3.1 Definisi

Sistem basis data adalah suatu sistem menyusun dan mengelola recordrecord menggunakan komputer untuk menyimpan atau merekam serta memelihara data opersional lengkap sebuah organisasi/perusahaan sehingga mampu menyediakan informasi yang optimal yang diperlukan pemakai untuk proses mengambil keputusan.

Untuk mengelola basis data diperlukan perangkat lunak yang disebut DBMS. DBMS adalah perangkat lunak sistem yang memungkinkan para pemakai membuat, memelihara, mengontrol, dan mengakses basis data dengan cara yang praktis dan efisien.

2.3.2 Tahap Perancangan Basis Data

Perancangan basis data merupakan langkah untuk menentukan basis data yang diharapkan dapat mewakili seluruh kebutuhan pengguna. Perancangan basis data terdiri atas perancangan basis data secara konseptual, perancangan basis data secara logis, dan perancangan basis data secara fisis.

Beberapa komponen yanng terdapat pada perancangan basis data secara konseptual antara lain:

1. Entitas

   Entitas terkadang disebut tipe entitas atau kelas entitas. Entitas adalah objek yang dapat dibedakan dari objek-objek lainnya.

2. Atribut

   Atribut adalah item data yang menjadi bagian dari suatu entitas. Istilah lain dari atribut adalah property.

3. Hubungan

   Hubungan adalah asosiasi atau kaitan antara dua entitas.

4. Kekangan

   Kekangan digunakan untuk melindungi integritas data (misalnya, melindungi kesalahan sewaktu pengisian data.

5. Domain

   Domain adalah himpunan yang berlaku bagi suatu atribut. Kekangan domain mendefinisikan nama, tipe, format, panjang, dan nilai masing-masing item data.

6. Integritas Referensial

   Integritas Referensial adalah aturan-aturan yang mengatur hubungan antara kunci primer dengan kunci tamu milik table-tabel yang berbeda dalam suatu basis data relasional untuk menjaga konsistensi data.

2.4 Metode Pendekatan Berorientasi Objek

Dalam suatu proses pengembangan sistem informasi, analisis dan perancangan merupakan terminologi yang sudah sangat tua. Sebenarnya pada saat masalah ditelusuri dan spesifikasi dinegoisasikan, dapat dikatakan kita berada pada tahap perancangan. Merancang adalah menemukan suatu cara untuk menyelesaikan masalah, salah satu *tool*/model untuk merancang *software* yang berorientasi objek (*object oriented*) adalah UML (FOW[16]).

2.4.1 Konsep Objek

Objek dalam "*software analysis & design"* adalah sesuatu yang berupa konsep (*concept*), benda (*thing*), dan sesuatu yang membedakannya dengan lingkungannya. Objek biasanya adalah kata benda, namum dalam konteks OOP objek bukan saja yang bisa dilihat atau diraba, tapi objek dapat pula merupakan sesuatu yang abstrak yang hidup di dalam sistem seperti *file*, tabel, *database, event, system messages* .

Objek dapat dikenali dari keadaannya dan juga operasinya, sebagai contoh sebuah mobil dikenali dari warnanya, bentuknya, sedangkan manusia dari suaranya, namanya, serta ciri-ciri yang akan membedakan objek tersebut dari objek lainnya.

Alasan mengapa pada saat ini memilih pendekatan berorientasi objek (*object oriented*) dalam pengembangan sistem pakar, pertama adalah *scalability* dimana objek lebih mudah dipakai untuk menggambarkan sistem yang besar dan komplek. Kedua, *dynamic modeling*, dimana objek dapat dipakai untuk pemodelan sistem dinamis dan *real time* .

2.4.2 Pengembangan Berorinetasi Objek

Object oriented development adalah suatu cara untuk pengembangan perangkat lunak dan sistem informasi berdasarkan abstraksi objek-objek yang ada di dunia nyata. (Brooks 1987, dikutip dari buku Object Oriented Modeling And Design, tulisan James Rumbaugh, dkk)mengemukakan bagian tersulit dari pengembangan perangkat lunak dan/atau sistem pakar adalah tahapan analisis dimana kita harus menganalisis masalah yang sangat rumit, yang kita jumpai di dunia nyata serta melakukan abstraksi terhadap masalah itu, kemudian melakukan perancangan agar kelak dapat diimplementasikan dengan cepat serta akurat pada komputer (baik secara perangkat keras maupun perangkat lunak) menjadi suatu sistem/perangkat lunak yang sesuai kebutuhan serta harapan pengguna.

2.4.3 Teknik Dasar OOA/D *(Object-Oriented Analysis/Design)*

Dalam dunia pemodelan, metodologi berorientasi objek walaupun terikat kaidah-kaidah standar, namun teknik pemilihan objek tidak terlepas pada subyektifitas *software analyst* dan *designer*. Beberapa objek akan diabaikan dan beberapa objek menjadi perhatian untuk diimplementasikan di dalam sistem, hal ini sah-sah saja karena kenyataan bahwa suatu permasalahan sudah tentu memiliki lebih dari satu solusi.

Ada 3 (tiga) teknik/konsep dasar dalam OOA/D, yaitu pemodulan (*encapsulation*), penurunan (*inheritance)* dan *polymorphism.*

1. *Encapsulation*

   Encapsulation (pengkapsulan) merupakan dasar untuk pembatasan ruang lingkup program terhadap data yang diproses.

2. *Inheritance*

   Inheritance (*pewarisan*) adalah teknik yang menyatakan bahwa anak (turunan) dari objek akan mewarisi data/atribut dan metoda dari induknya langsung.

3. *Polymorfhism*

   Polimorfisme adalah konsep yang menyatakan bahwa sesuatu yang sama dapat mempunyai bentuk dan perilaku yang berbeda.

2.4.4 Pengenalan UML

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

UML merupakan bahasa pemodelan yang memiliki pembendaharaan kata dan cara untuk mempresentasikan secara fokus pada konseptual dan fisik dari suatu sistem. Contoh untuk sistem *software* yang *intensive* membutuhkan bahasa yang menunjukan pandangan yang berbeda dari arsitektur sistem, ini sama seperti menyusun/mengembangkan *software develpoment life cycle*. Dengan UML akan memberitahukan kita bagaimana untuk membuat dan membaca bentuk model yang baik, tetapi UML tidak dapat memberitahukan model apa yang akan

dibangun dan kapan akan membangun model tersebut. Ini merupakan aturan dalam *software development process* .

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*.

Dalam kerangka spesifikasi, UML menyediakan model-model yang tidak mendua-arti (ambigu), serta lengkap. Secara khusus, UML menspeifikasi langkah-langkah penting dalam pengambilan keputusan analisis, perancangan, serta implementasi dalam sistem yang sangat bernuansa perangkat lunak (*software intensive system*).

2.4.5 Bagian-Bagian dari UML

Bagian–bagian utama dari UML adalah *view, diagarams, model element, dan general mechanism.*

1. *View*

   View digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda. *View* bukan melihat grafik, tapi merupakan suatu abstraksi yang berisi sejumlah diagram. Beberapa jenis *view* dalam UML antara lain *use case view, logical view, component view, concurrences view,* dan *deployment view*.

a. *Use case view*

Mendeskripsikan fungsionalitas sistem yang seharusnya dilakukan sesuai dengan yang diinginkan *external actors. Actor* yang berinterakasi dengan sistem dapat berupa *user* atau sistem lainnya. *View* ini digambarkan dalam *use case diagram* dan kadang–kadang dengan *activity diagrams. View* ini digunakan terutama untuk pelanggan, perancang (*designer*), pengembang *(developer*), dan penguji sistem (*tester*).

b. *Logical view*

Mendeskripsikan bagaimana fungsionalitas dari sistem, struktur statis (*class, object*, dan r*elationship*) dan kolaborasi dinamis yang terjadi ketika *object* mengirim pesan ke *object* lain dalam suatu fungsi tertentu. *View* ini digambarkan dalam *class diagram* untuk struktur statis dan dalam *state sequence*, *collaboration,* dan *activity diagrams* untuk model dinamisnya. *View* ini digunakan untuk perancang (*designer*) dan pengembang (*developer*).

c. *Component view*

Mendiskripsikan implementasi dan ketergantungan modul. Komponen yang merupakan tipe lainnya dari *code module* diperlihatkan dengan struktur dan ketergantungannya juga alokasi sumber daya komponen dan informasi *administrative* lainnya. *View* ini digambarkan dalam *component view* dan digunakan untuk pengembang (*developer*).

d. *Concurrency view*

Membagi sistem ke dalam proses dan *processor*. *View* ini digambarkan dalam diagram dinamis (*state, sequence, collaboration,* dan *activity diagram*) dan diagram implementasi (*component* dan *deployment diagrams*) serta digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji (*tester*).

e. *Deployment view*

Mendeskripsikan fisik dari sistem seperti komputer dan perangkat lain serta bagaimana hubungan dengan yang lainnya. *View* ini digambarkan dalam *deplovment diagram* dan digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji (*tester*).

2. *Diagrams*

Setiap sistem yang kompleks seharusnya bisa dipandang dari sudut yang berbeda-beda sehingga kita bisa mendapatkan pemahaman secara menyeluruh. Untuk upaya tersebut UML menyediakan setidaknya sembilan jenis diagram. Diagram-diagram ini tidak mutlak harus digunakan semuanya, hanya dibuat sesuai dengan kebutuhan. Pada pemodelan dengan UML juga dimungkinkan menggunakan diagram-diagram lain sejauh itu memang diperlukan untuk mendapatkan pemahaman mendalam tentang suatu sistem atau suatu perangkat lunak. Diagram-diagram yang disediakan UML antara lain :

a. *Class diagram*

Diagram ini memperlihatkan himpunan kelas, antarmuka, kolaborasi serta relasi. Diagram ini umumnya dijumpai pada pemodelan berorientasi objek. Meskipun bersifat statis sering pula diagram kelas memuat kelas-kelas aktif.

b. *Object diagram*

Diagram ini memperlihatkan objek-objek serta relasi antar objek. Diagram objek memperlihatkan istansiasi statis dari segala sesuatu yang dijumpai pada diagram kelas.

c. *Use Case diagram*

Diagram ini memperlihatkan himpunan *Use Case* dan aktor-aktor. Diagram ini sangat penting terutama untuk mengorganisasi dan memodelkan perilaku-perilaku dari suatu sistem yang dibutuhkan serta diharapkan pengguna.

d. *Sequence diagram*

Diagram ini berfungsi sebagai interaksi yang menekankan pada pengiriman pesan (*message*) dalam suatu waktu tertentu.

e. *Collaboration diagram*

Diagram interaksi yang menekankan organisasi struktural dari objek-objek yang menerima serta mengirim pesan (*message*).

f. *State diagram*

Menggambarkan semua *state* pada sistem, memuat *state*, transisi, *event* serta aktifitas. Diagram ini penting untuk memperlihatkan sifat

dinamis dari antarmuka, kelas serta kolaborasi terutama pada pemodelan sistem-sistem yang reaktif.

g. *Aktivity diagram*

Diagram ini adalah tipe khusus dari diagram *state* yang memperlihatkan aliran dari suatu aktifitas ke aktifitas lainnya dalam suatu sistem. Diagram ini penting bagi pemodelan fungsi-fungsi dalam suatu sistem dan memberikan tekanan pada aliran kendali antarobjek.

h. *Component Diagram*

Diagram ini memperlihatkan organisasi serta ketergantungannya pada komponen-komponen yang telah ada sebelumnya. Diagram ini berhubungan dengan diagram kelas dimana komponen secara tipikal dipetakan ke dalam satu atau lebih kelas, antarmuka serta kolaborasi.

i. *Deployment Diagram*

Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan (*run-time*). Diagram ini memuat simpul-simpul (*node*) beserta komponen-komponen yang ada di dalamnya. *Deployment diagram* berhubungan erat dengan *component diagram* dimana *deployment diagram* memuat satu atau lebih komponen. Diagram ini berguna saat aplikasi akan diterapkan pada banyak komputer (*distributed computing*).

2.5 Bahasa Pemrograman *Borland Delphi 6.0*

Dalam pembuatan sebuah program, Delphi menggunakan sistem yang disebut RAD (*Rapid Application Development*). Sistem ini memanfaatkan bahasa pemrograman visual yang mempermudah bagi pemakainya mendesain tampilan program (*user interface*). Cara ini sangat bermanfaat untuk membuat program yang bekerja dalam sistem Windows yang memang tampilan layarnya lebih rumit dibandingkan dengan sistem DOS dulu.

Program Delphi dikembangkan dengan bahasa Pascal dan bekerja dalam lingkungan *Windows 95* atau lebih. Delphi menjadikan pemakai tidak perlu membuat tampilan kotak layar, dialog, perangkat kontrol yang rumit dan menyita waktu dalam penulisan program. Beberapa keunggulan lainnya adalah kecepatan eksekusinya yang cepat, serta kemudahan akses berbagai format database.

Borland Delphi memiliki keunggulan dibandingkan dengan aplikasi pemrograman visual berbasis Windows lainnya, diantaranya:

1) Borland Delphi memiliki RAD (*Rapid Application Development*), yaitu perangkat pengembang yang mampu dengan mudah dan cepat menghasilkan program aplikasi.
2) Borland Delphi dapat membangun suatu program aplikasi dengan tatanan GUI (*Graphical User Interface*), yaitu program aplikasi yang menggunakan perantara untuk berinteraksi dengan pemakai (*user*).
3) Borland Delphi dapat dipakai untuk merancang program aplikasi sesuai dengan yang kita inginkan, serta mudah dan cepat di akses.

2.6 *Microsoft Access* 2003

Microsoft Access merupakan suatu program pengolahan database yang canggih, yang digunakan untuk mengolah berbagai jenis data dengan pengoperasian yang mudah. Banyak kemudahan yang di peroleh jika bekerja dengan Microsoft Access. Diantaranya dapat melakukan proses penyortiran, pengaturan data, pembuatan table data serta pembuatan laporan.

Microsoft Access adalah suatu aplikasi yang dapat membantu kita membuat sebuah aplikasi database dalam waktu yang relatif singkat. Bagian dari Microsoft Access adalah :

a. Table digunakan untuk menyimpan data
b. Query digunakan untuk memanipulasi data
c. Form digunakan untuk fronted aplikasi. Biasanya untuk menampilkan data, menambah data, dll.
d. Report digunakan untuk membuat laporan
e. Macro digunakan untuk melakukan satu atau beberapa fungsi.