RANCANG BANGUN SISTEM PAKAR UNTUK PERBAIKAN KECEPATAN DAN KEGAGALAN KONEKSI PERALATAN EKSTERNAL PADA PERSONAL KOMPUTER

I Made Sukarsa[1] , Ni Wayan Wisswani[2]
[1)] Staff Pengajar Teknik Elektro, Fakultas Teknik, Universitas Udayana
Kampus Bukit Jimbaran, Bali, 80361
[2)] Jurusan Teknik Elektro Politeknik Negeri Bali
e_arsa@yahoo.com

Abstrak

Sistem pakar pada penelitian ini merupakan sistem pakar berbasis web yang menghadirkan solusi dalam mengatasi kerusakan komputer khususnya untuk perbaikan kecepatan dan kegagalan koneksi peralatan eksternal. Sistem menyediakan fasilitas troubleshooting untuk membantu user dalam mengidentifikasi permasalahan komputernya. Proses identifikasi ini dilakukan melalui interaksi tanya-jawab antara sistem dengan pemakai. Sistem juga menyediakan fasilitas edit pengetahuan yang dapat digunakan oleh Knowledge Engineer dalam melakukan perubahan data pada basis pengetahuan. Metode penelusuran solusi yang digunakan pada sistem ini adalah adalah penelusuran runut maju (forward chainning). Aturan pada basis pengetahuan dimodelkan sebagai tree dengan memanfaatkan database MySQL serta bahasa pemrograman PHP.

Pemanfaatan database untuk menyimpan basis pengetahuan dari sistem pakar akan mempermudah dalam pembuatan fasilitas penambahan pengetahuan. Dengan adanya fasilitas penambahan pengetahuan, perubahan aturan pada basis pengetahuan dan pengembangan sistem melalui akuisisi pengetahuan yang baru dapat langsung dilakukan tanpa harus membongkar sistem yang sudah jadi. Hal ini akan memungkinkan sistem menjadi tetap up to date.

Kata kunci : *Sistem Pakar, Kerusakan komputer.*

1. PENDAHULUAN

Sistem pakar untuk pada penelitian ini merupakan sistem berbasis web yang menghadirkan solusi dalam mengatasi kerusakan komputer. Sistem menyediakan fasilitas *troubleshooting* untuk membantu *user* dalam mengidentifikasi permasalahan komputernya. Proses identifikasi ini dilakukan melalui interaksi tanya-jawab antara sistem dengan pemakai. Sistem juga menyediakan fasilitas edit pengetahuan yang dapat digunakan oleh *Knowledge Engineer* dalam melakukan perubahan data pada basis pengetahuan. Metode penelusuran solusi yang digunakan pada sistem ini adalah adalah penelusuran runut maju (*forward chainning*). Aturan pada basis pengetahuan dimodelkan sebagai *tree* dengan memanfaatkan database MySQL serta bahasa pemrograman PHP.

Pemanfaatan database untuk menyimpan basis pengetahuan dari sistem pakar akan mempermudah dalam pembuatan fasilitas penambahan pengetahuan. Dengan adanya fasilitas penambahan pengetahuan, perubahan aturan pada basis pengetahuan dan pengembangan sistem melalui akuisisi pengetahuan yang baru dapat langsung dilakukan tanpa harus membongkar sistem yang sudah jadi. Hal ini akan memungkinkan sistem menjadi tetap *up to date.*

2. TINJAUAN PUSTAKA

2.1 Sistem Pakar

Sistem pakar adalah sebuah perangkat lunak komputer yang memiliki basis pengetahuan untuk domain tertentu dan menggunakan penalaran inferensi menyerupai seorang pakar dalam memecahkan masalah. Sedangkan definisi lain dari sistem pakar adalah sistem berbasis komputer yang menggunakan pengetahuan, fakta, dan teknik penalaran dalam memecahkan masalah yang biasanya hanya dapat dipecahkan oleh seorang pakar dalam suatu bidang tertentu.

Sistem pakar menggabungkan pengetahuan dan penelusuran data untuk memecahkan masalah yang secara normal memerlukan keahlian manusia. Tujuan dari Sistem Pakar sebenarnya bukan untuk menggantikan peran manusia, tetapi untuk mensubtitusikan pengetahuan manusia kedalam bentuk sistem, sehingga dapat digunakan oleh orang banyak (Nugroz, 2007).

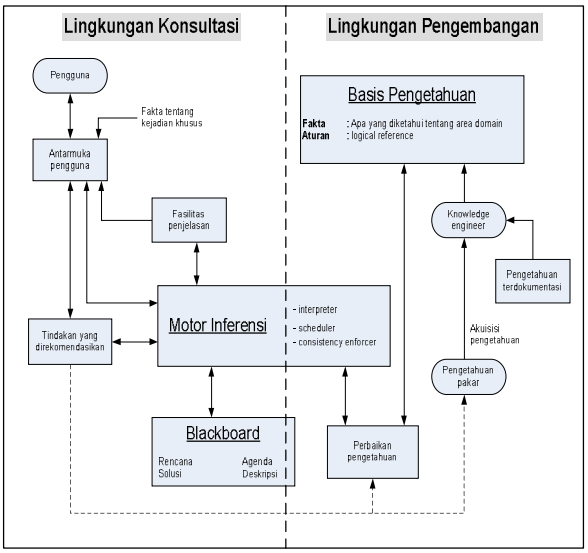
2.2 Struktur Sistem Pakar

Menurut Turban (2005), sistem pakar dapat ditampilkan dengan dua lingkungan, yaitu : lingkungan pengembangan dan lingkungan konsultasi (*runtime*). Lingkungan pengembangan digunakan oleh *ES builder* untuk membangun komponen dan memasukkan pengetahuan ke dalam basis

pengetahuan. Lingkungan konsultasi digunakan oleh user nonpakar untuk memperoleh pengetahuan dan nasihat pakar. Lingkungan ini dapat dipisahkan setelah sistem lengkap.

Tiga komponen utama yang tampak secara virtual di setiap sistem pakar adalah basis pengetahuan, mesin inferensi, dan antarmuka pengguna. Sistem pakar yang berinteraksi dengan pengguna dapat pula berisi komponen tambahan sebagai berikut :

- Subsistem akuisisi pengetahuan
- *Blackboard* (tempat kerja)
- Subsistem penjelasan (*justifier*)
- Sistem perbaikan pengetahuan



Gambar 1. Struktur sistem pakar

Kebanyakan sistem pakar saat ini tidak berisi komponen perbaikan pengetahuan. Deskripsi singkat tiap komponen dapat dilihat pada bagian berikut :

1. Subsistem Akuisisi Pengetahuan

Akuisisi pengetahuan adalah akumulasi, transfer, dan transformasi keahlian pemecahan masalah para pakar atau sumber pengetahuan terdokumentasi ke program komputer, untuk membangun atau memperluas basis pengetahuan. Sumber pengetahuan potensial antara lain pakar manusia, buku teks, dokumen multimedia, *database*, laporan riset khusus, dan informasi yang terdapat dalam *web*.

Mendapatkan pengetahuan dari pakar adalah tugas kompleks yang sering menimbulkan kemacetan dalam konstruksi sistem pakar. Dalam membangun sistem yang besar, seseorang memerlukan *knowledge engineer* atau pakar elisitasi pengetahuan untuk berinteraksi dengan satu atau lebih pakar manusia. Biasanya *knowledge engineer* membantu pakar memnyusun area persoalan dengan menginterpretasikan dan mengintegrasikan jawaban manusia, menyusun analogi, mengajukan contoh pembanding dan menjelaskan kesulitan konseptual.

2. Basis Pengetahuan *(Knowledge Base)*

Basis pengetahuan merupakan representasi pengetahuan dari seorang pakar yang diperlukan untuk memahami, memformulasikan dan memecahkan masalah. Terdiri dari dua elemen dasar, yaitu :

- Fakta yang berupa informasi tentang situasi permasalahan, teori dari area permasalahan atau informasi tentang objek.
- Spesial heuristik yang merupakan informasi tentang cara bagaimana membangkitkan fakta baru dari fakta yang sudah diketahui. Dalam sistem pakar berbasis *rule*, bagian ini berupa *rules*.

Knowledge base adalah jantung sebuah sistem pakar. Bagian ini adalah totalitas keahlian pakar yang telah disarikan dan diformat ke dalam external memory komputer. Sampai saat ini terdapat berbagai cara representasi pengetahuan yang telah dikenal, misalnya :

a. Rule-Based Knowledge

Pengetahuan direpresentasikan dalam suatu bentuk fakta (*facts*) dan aturan (*rules*). Bentuk representasi ini terdiri atas premise dan kesimpulan. Pada penalaran berbasis aturan, pengetahuan dipersentasikan dengan menggunakan aturan berbentuk : IF-THEN. Bentuk ini digunakan apabila kita memiliki sejumlah pengetahuan pakar pada suatu permasalahan tertentu, dan pakar dapat menyelesaikan masalah tersebut secara berurutan. Disamping itu, bentuk ini juga digunakan apabila dibutuhkan penjelasan tentang langkah-langkah pencapaian solusi.

b. Case-Base Reasoning

Pada penalaran berbasis kasus (*cases*), basis pengetahuan berisi solusi-solusi yang telah dicapai sebelumnya, kemudian diturunkan suatu solusi untuk keadaan sekarang (fakta yang ada). Bentuk ini digunakan apabila *user* menginginkan untuk mengetahui lebih banyak lagi pada kasus-kasus yang hampir sama (mirip). Selain itu, bentuk ini juga digunakan apabila telah dimiliki sejumlah situasi atau kasus tertentu dalam basis pengetahuan atau dapat diartikan pengetahuan direpresentasikan dalam bentuk kesimpulan kasus.

c. Frame-Based Knowledge

Pengetahuan direpresentasikan dalam suatu bentuk hirarki atau jaringan *frame*.

d. Object-Based Knowledge

Pengetahuan direpresentasikan sebagai jaringan dari objek-objek. Objek adalah elemen data yang terdiri dari data dan metode (proses).

3. Mesin Inferensi *(Inference Engine)*

Inference engine merupakan otak dari sistem pakar, bagian ini mengandung mekanisme fungsi berpikir dan pola-pola penalaran sistem yang digunakan oleh seorang pakar. Mekanisme ini akan menganalisa suatu masalah tertentu dan kemudian mencari jawaban atau kesimpulan yang terbaik. Dari fakta-fakta yang diperoleh selama proses tanya-jawab dengan *user*, serta aturan-aturan yang tersimpan di *knowledge base*, *inference engine* dapat menarik suatu kesimpulan dan memberikan rekomendasi atau saran yang diharapkan oleh *user*.

Ada dua metode dasar yang bisa digunakan oleh mesin inferensi dalam mencari kesimpulan untuk mendapatkan solusi bagi permasalahan yang dihadapi sistem pakar, yaitu runut maju (*forward chaining*) dan runut balik (*backward chaining*). Berikut ini penjelasan mengenai kedua metode pencarian tersebut (Yunanto, 2003) :

a. Runut maju (*Forward chaining*)

Runut maju merupakan metode pencarian yang memulai proses pencarian dari sekumpulan data atau fakta, dari fakta-fakta tersebut dicari suatu kesimpulan yang menjadi solusi dari permasalahan yang dihadapi. Mesin inferensi mencari kaidah-kaidah dalam basis pengetahuan yang premisnya sesuai dengan fakta-fakta tersebut, kemudian dari aturan-aturan tersebut diperoleh suatu kesimpulan. Runut maju memulai proses pencarian dengan data sehingga strategi ini disebut juga *data-driven*.

b. Runut balik (*Backward chaining*).

Runut balik merupakan metode pencarian yang arahnya kebalikan dari runut maju. Proses pencarian dimulai dari tujuan, yaitu kesimpulan yang menjadi solusi dari permasalahan yang dihadapi. Mesin inferensi mencari aturan-aturan dalam basis pengetahuan yang kesimpulannya merupakan solusi yang ingin dicapai, kemudian dari aturan-aturan yang diperoleh, masing-masing kesimpulan dirunut balik jalur yang mengarah ke kesimpulan tersebut. Jika informasi-informasi atau nilai dari atribut-atribut yang mengarah ke kesimpulan tersebut sesuai dengan data yang diberikan maka kesimpulan tersebut merupakan solusi yang dicari, jika tidak sesuai maka kesimpulan tersebut bukan merupakan solusi yang dicari. Runut balik memulai proses pencarian dengan suatu tujuan sehingga strategi ini disebut juga *goal-driven*.

4. Antarmuka Pengguna *(User Interface)*

Merupakan bagian dari sistem pakar yang berfungsi sebagai pengendali *input-output. User interface* melayani *user* selama proses konsultasi mulai dari tanya-jawab untuk mendapatkan fakta-fakta yang dibutuhkan oleh *inference engine* sampai menampilkan *output* yang merupakan kesimpulan atau rekomendasi yang dihasilkan oleh *inference engine*.

5. Tempat Kerja *(Blackboard)*

Blackboard adalah area kerja memori yang disimpan sebagai *database* untuk deskripsi persoalan terbaru yang ditetapkan oleh data input, digunakan juga untuk perekaman hipotesis dan keputusan sementara. Tiga tipe keputusan dapat direkam dalam *blackboard*, antara lain : rencana (bagaimana mengatasi persoalan), agenda (tindakan potensial sebelum eksekusi), dan solusi (hipotesis kandidat dan arah tindakan alternatif yang telah dihasilkan sistem sampai dengan saat ini).

6. Subsistem Penjelasan *(Justifier)*

Bagian yang harus siap memberikan penjelasan saat *user* perlu mengetahui apakah alasan diberikannya sebuah solusi. Bagian ini secara konkrit membedakan sebuah sistem pakar dengan sistem aplikasi yang biasa, karena pada pemrograman konvensional tidaklah biasa sebuah sistem menyediakan informasi tambahan mengapa atau dari mana sebuah solusi diperoleh.

Bagian ini mempunyai kemampuan untuk menelusuri konklusi dan menerangkan tingkah laku sistem pakar dengan menjawab pertanyaan-pertanyaan seperti :

- Mengapa pertanyaan tersebut diajukan oleh sistem pakar?
- Bagaimana atau darimana konklusi tersebut diperoleh?
- Mengapa alternatif tersebut ditolak?

Pada sistem pakar berbasis aturan, biasanya penjelasan ini dilakukan dengan cara memperlihatkan aturan-aturan yang digunakan. Fasilitas ini penting untuk menambah rasa percaya user pada hasil *output* program sistem pakar yang digunakannya.

7. **Sistem Perbaikan Pengetahuan *(Knowledge-base editor)***

Bagian yang digunakan untuk menambah, menghapus atau memperbaiki basis pengetahuan. Bagian ini tidaklah mutlak, karena mayoritas sistem pakar berbasis pengetahuan dalam format *text-file*, sehingga bagian ini dapat digantikan dengan berbagai *word processor* yang tersedia. Namun demikian bila sistem pakar dituntut untuk memiliki kemampuan *machine learning* dari pengalaman konsultasinya, bagian ini menjadi sangat vital.

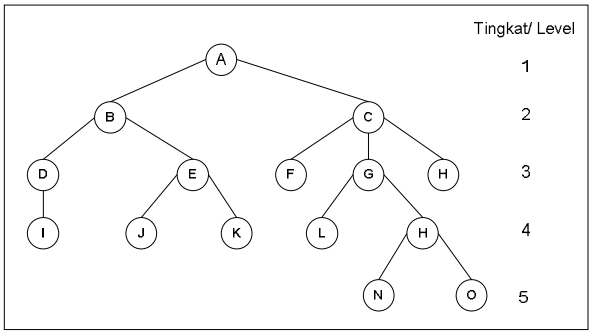
8. Fakta Tentang Kejadian Khusus

Bagian ini hanya diperlukan saat data yang telah dimiliki pemakai (*file database* atau *spreadsheet)* diperlukan sebagai referensi untuk menarik kesimpulan.

2.3 Struktur Pohon

Pohon atau *tree* adalah salah satu metode yang dapat digunakan untuk membuat suatu pemodelan. Struktur ini memiliki sifat-sifat atau ciri-ciri khusus, dan biasanya digunakan untuk menggambarkan hubungan yang bersifat hirarkis antara elemen-elemen yang ada. Contoh paling sederhana yang bisa dilihat dalam kehidupan sehari-hari adalah struktur organisasi dari suatu perusahaan serta pada silsilah keluarga.

Secara sederhana pohon dapat didefinisikan sebagai kumpulan elemen yang salah satu elemennya disebut dengan akar (*root*), dan sisa elemen yang lain (yang disebut simpul) terpecah menjadi sejumlah himpunan yang saling tidak berhubungan satu sama lain, yang disebut dengan subpohon (*subtree*), atau juga disebut dengan cabang. Jika dilihat pada setiap subpohon, maka subpohon inipun mempunyai akar dan subpohonnya masing-masing. Dengan demikian pohon ini merupakan salah satu contoh dari bentuk rekursif. Untuk lebih jelasnya dapat diperhatikan gambar berikut.



Gambar 2. Contoh Pohon dengan 15 Simpul

Simpul (*node* atau *vertex*) adalah elemen pohon yang berisi informasi/data dan penunjuk percabangan. Pohon pada gambar di atas berisi 15 simpul yang berisi informasi berupa huruf A, B, C, D dan seterusnya sampai huruf O lengkap percabangan nya, dengan akarnya yang berisi huruf A.

Hubungan antara satu simpul dengan simpul lain bisa dianalogikan dalam sebuah keluarga, yaitu ada anak, bapak, paman dan lain-lain. Sesuai dengan gambar di atas, simpul A adalah bapak dari simpul B, dan C; dengan demikian simpul B dan C ini bersaudara. Simpul D dan E adalah anak dari simpul B. Simpul C adalah paman dari simpul D dan E.

Tingkat *(level)* suatu simpul ditentukan dengan pertama kali menentukan akar sebagai bertingkat 1. Jika suatu simpul dinyatakan sebagai tingkat N, maka simpul-simpul yang merupakan anaknya dikatakan berada dalam tingkat N + 1. Gambar 2 menunjukkan contoh pohon lengkap dengan tingkat setiap simpulnya.

Selain tingkat, juga dikenal istilah derajad *(degree)* dari suatu simpul. Derajad suatu simpul dinyatakan sebagai banyaknya anak atau turunan dari simpul tersebut. Sebagai contoh, di atas simpul A mempunyai derajad 2, simpul B mempunyai derajad 2 dan simpul C berderajad 3. Simpul-simpul F, H, I, J, K, L, N, O yang semuanya berderajad nol, disebut dengan daun (*leaf*). Daun juga sering disebut dengan simpul luar *(external node)*, sehingga simpul lain, kecuali akar, juga sering disebut dengan simpul dalam *(internal node)*.

Tinggi *(height)* atau kedalaman *(depth)* dari suatu pohon adalah tingkat maksimum dari simpul dalam pohon tersebut dikurangi dengan 1. Dengan demikian, pohon pada gambar di atas yang berakar pada A mempunyai tinggi atau kedalaman 4.

2.4 Fungsi Rekursif

Rekursif adalah salah satu metode dalam dunia matematika dimana definisi sebuah fungsi mengandung fungsi itu sendiri. Dalam dunia pemrograman, rekursi diimplementasikan dalam sebuah fungsi yang memanggil dirinya sendiri. Contoh fungsi rekursif misalnya adalah fungsi pangkat, faktorial, dan barisan fibonacci.

Ide dasar dalam memecahkan suatu masalah dengan rekursif adalah sebagai berikut :

- Tentukan kasus penyetop atau kasus dasar di mana pemanggilan rekursif tidak lagi diperlukan (karena solusinya sudah diperoleh).
- Terapkan suatu langkah untuk menggiring kasus kompleks ke kasus penyetopnya dengan metode yang mencerminkan fungsinya.

Berikut adalah perbandingan keuntungan dan kelebihan fungsi rekursif dibandingkan iteratif (Widiarta, 2007) :

Tabel 1. Perbandingan rekursif dengan iteratif

Rekursif	Iteratif
Kode program biasanya lebih ringkas dan mudah dipahami	Kode program lebih panjang, untuk beberapa kasus solusi iteratif lebih sulit diterapkan
Membutuhkan alokasi memori yang besar	Relatif lebih kecil alokasi memorinya
Tidak cocok ketika kinerja tinggi diperlukan, karena terjadi *overhead* pemanggilan fungsi dalam jumlah yang relatif besar	Cocok diterapkan ketika kinerja aplikasi harus diterapkan (hanya ada satu kali pemanggilan fungsi)

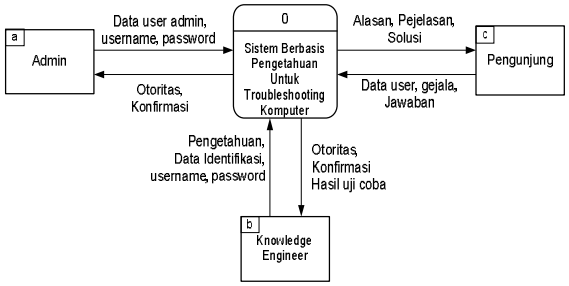
3. METODE PERANCANGAN SISTEM

3.1 Perancangan *Data Flow Diagram* (DFD)

Untuk perancangan *data flow diagram* (DFD), hal-hal yang diperlukan antara lain : mengidentifikasi semua kesatuan luar yang terlibat dalam sistem, menggambarkan diagram konteks, bagan berjenjang kemudian dilanjutkan dengan penggambaran DFD yang dimulai dari level 0.

3.1.1 Penggambaran Diagram Konteks

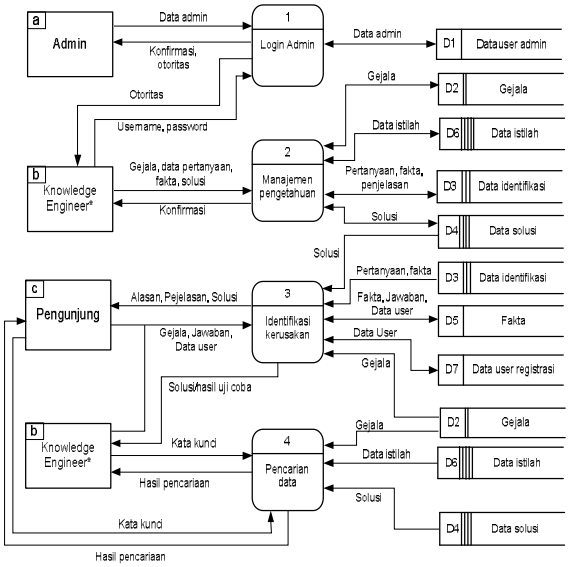
Pemodelan hubungan sistem dengan kesatuan luarnya dapat dilihat pada diagram konteks. Diagram konteks dari sistem pakar untuk *troubleshooting* komputer dapat dilihat pada gambar berikut :



Gambar 3. Diagram berjenjang sistem untuk *troubleshooting* komputer

3.1.2 *Data Flow Diagram* (DFD) Level 0

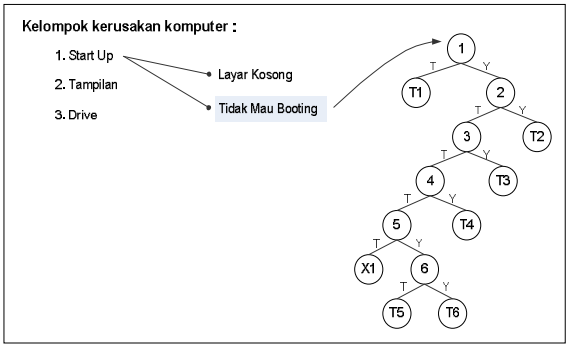
DFD akan menggambarkan sistem yang akan direncanakan dan nantinya dapat digunakan sebagai landasan dalam pengembangan sistem lebih lanjut. Berikut ini adalah DFD dari sistem yang direncanakan :



Gambar 4. DFD level 0 sistemuntuk *troubleshooting* komputer

3.2 Pemodelan Basis Pengetahuan dengan Pohon

Basis pengetahuan pada sistem ini merupakan tempat tersimpannya kumpulan aturan yang memiliki keterkaitan satu sama lain. Hubungan antar aturan tersebut dapat dimodelkan dengan menggunakan struktur *tree.* Berikut ini merupakan salah satu contoh tree pada basis penegtahuan.



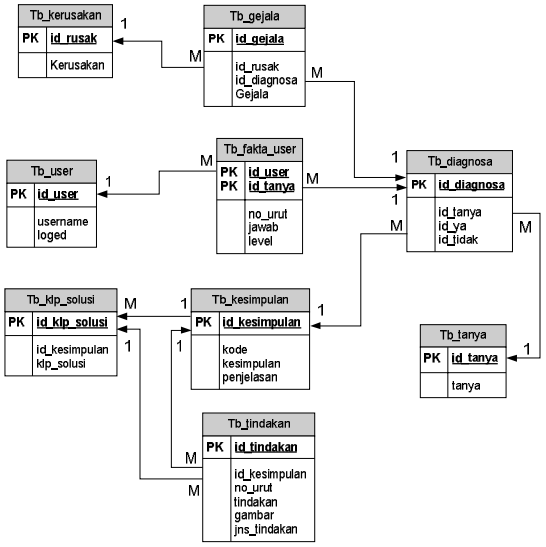
Gambar 5. Pemodelan basis pengetahuan dengan pohon

Keterangan untuk Gambar 5.

1	:	Apakah sistem melewatkan pengujian memori
2	:	Apa anda mendengar lebih dari 1 bunyi beep
3	:	Apakah anda dihentikan sebuah layar password atau pesan error CMOS
4	:	apakah sistem booting dari drive yang salah
5	:	apakah anda mencoba booting dari sebuah hard drive
6	:	Apakah anda memperoleh sebuah pesan kesalahan sebelum starting windows
T1	:	Komputer saya gagal melakukan pengujian memori
T2	:	Komputer berbunyi beep bukannya booting
T3	:	Komputer memberitahu tentang kesalahan CMOS atau meminta sebuah password
T4	:	Komputer booting dari disk drive yang salah
T5	:	Saya melihat pesan kesalahan sebelum sistem menampilkan pesan Starting Windows
T6	:	Sistem saya memperlihatkan pesan Starting Windows, namun kemudian memperlihatkan pesan kesalahan
X1	:	Menuju ke kelompok kerusakan “Drive” dengan gejala “Masalah floppy”

3.3 Hubungan Antar Tabel

Hubungan antar tabel pada rancangan sistem yang akan dibuat adalah seperti dalam gambar berikut :

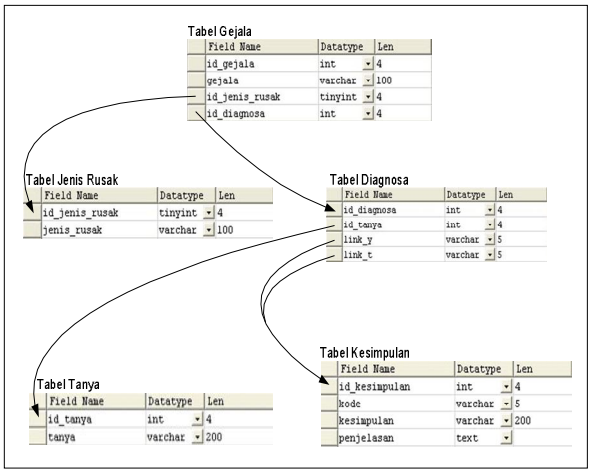


Gambar 6. Hubungan antar tabel

4. PEMBAHASAN DAN ANALISIS SISTEM

4.1 Pemodelan *Tree* pada *Database*

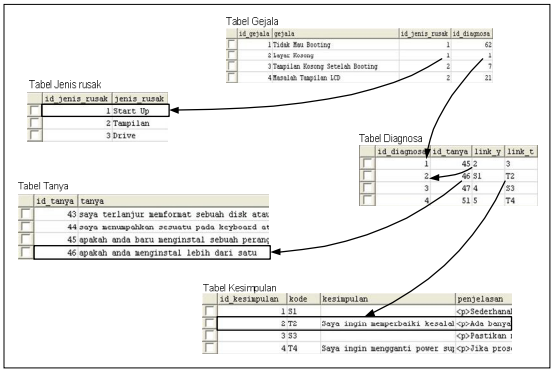
Tree atau pohon pada sistem pakar ini digunakan untuk menggambarkan hubungan antara aturan yang satu dengan aturan yang lain yang terdapat pada basis pengetahuan. Struktur *tree* dibangun menggunakan *database* MySQL yang nantinya ditampilkan dengan fungsi yang dibuat dengan bahasa pemrograman PHP. Berikut ini adalah struktur tabel yang digunakan untuk menyimpan basis pengetahuan.



Gambar 7. Struktur tabel dari basis pengetahuan

Semua aturan yang ada tersimpan pada tabel diagnosa. Sedangkan untuk detail dari aturan, baik pertanyaan gejala maupun kesimpulan masing-masing tersimpan pada tabel tanya dan tabel kesimpulan.

Gejala umum yang ada dikelompokkan ke dalam beberapa jenis kerusakan. Pengelompokan ini bertujuan agar pohon yang nantinya akan digunakan dalam proses identifikasi masalah menjadi ringkas dan efisien dalam membantu *user*. Masing-masing gejala umum memiliki satu pertanyaan awal yang diambil dari tabel diagnosa. Gambar 8 yang menjelaskan hubungan antar tabel yang disertai dengan contoh datanya.



Gambar 8. Hubungan antar tabel dalam basis pengetahuan

Dari gambar 8 terlihat jelas bagaimana hubungan antar tabel dalam basis pengetahuan. Setiap kategori kerusakan pada tabel jenis_rusak dapat memiliki beberapa gejala yang tersimpan pada tabel gejala. Sebagai contoh, kategori kerusakan komputer pada "Start up" memiliki dua gejala kerusakan yaitu "Tidak Mau Booting" dan "Layar Kosong" yang terdapat pada tabel gejala. Masing-masing gejala kerusakan ini memiliki satu pertanyaan awal yang diambil dari tabel diagnosa. Pada tabel diagnosa terdapat beberapa *field*, seperti : id_tanya, link_y dan link_t. *Field* id_tanya menyimpan id pertanyaan yang detailnya ada pada tabel tanya. Link_y menyimpan id aturan yang akan digunakan jika pertanyaan dijawab "iya", sebaliknya link_t menyimpan id aturan yang akan digunakan jika pertanyaan dijawab "tidak". Jika pada link_y atau link_t ditemukan kode yang diawali dengan huruf, berarti pada pertanyaan tersebut telah ditemukan solusi yang detailnya ada pada tabel kesimpulan. Pada tabel ini, kesimpulan atau solusi dikelompokkan menjadi 2 jenis yang ditandai dengan perbedaan huruf depan pada kodenya. Jika kode diawali dengan huruf "S" berarti solusi tersebut termasuk jenis solusi cepat/singkat, sedangkan jika diawali dengan huruf "T" berarti solusi tersebut merupakan solusi lengkap yang memiliki tahapan detail pada tabel lain.

Selanjutnya pohon dapat ditampilkan pada halaman HTML cukup dengan melakukan pemanggilan fungsi rekursif yang telah dibuat dengan menggunakan parameter id_diagnosa yang dimiliki oleh setiap gejala umum. Berikut adalah contoh tampilan pohon yang dimiliki oleh gejala umum "Tidak Mau *Booting*".

Gambar 9. Contoh tampilan pohon dari gejala umum

4.2 Proses Identifikasi Masalah

Pada sistem pakar untuk *troubleshooting* komputer ini terdapat fasilitas untuk membantu *user* dalam melakukan identifikasi permasalahan komputer yang dihadapi. Sebelum memulai proses *troubleshooting*, terlebih dahulu *user* harus melakukan registrasi pada sistem.

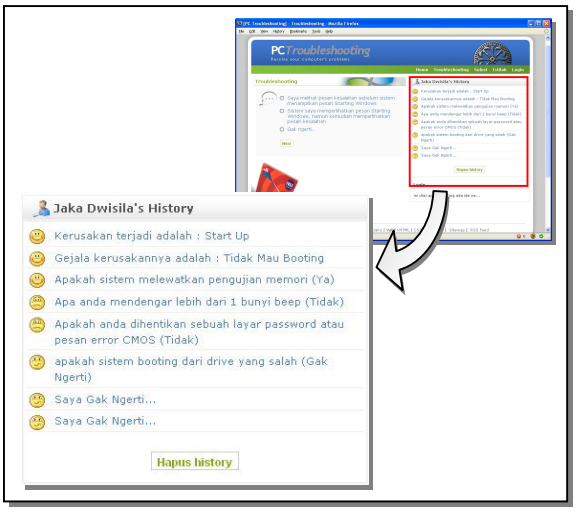
Gambar 10. Halaman registrasi *user*

Registrasi ini bertujuan untuk membedakan data *user* yang satu dengan *user* yang lain, sehingga memungkinkan beberapa *user* untuk melakukan *troubleshooting* pada saat yang bersamaan dan tentunya dari lokasi yang berbeda. Data *user* selanjutnya akan disimpan pada database *server* dan juga disimpan pada *client* sebagai *cookie.*

Setelah melakukan registrasi, *user* langsung diarahkan ke halaman *Troubleshooting* untuk memulai identifikasi atas permasalahan komputer yang dihadapi. Identifikasi masalah akan diawali dengan memilih kelompok kerusakan dan gejalanya, yang kemudian dilanjutkan dengan proses tanya jawab antara *user* dengan sistem hingga ditemukannya sebuah solusi. Semua aktivitas *user* pada proses ini akan disimpan ke *database* yang selanjutnya ditampilkan sebagai data *history*. Setiap awal memulai proses *troubleshooting* data *history* akan terlihat masih kosong.

Gambar 11. Halaman *Troubleshooting*

Data history digunakan untuk menampilkan pertanyaan beserta jawaban *user* yang telah dilalui sebelumnya, sehingga *user* dengan mudah dapat melihat apa yang telah dilakukan selama proses *troubleshooting* ini. Dengan ditampilkannya data *history* juga akan memberikan kesempatan bagi *user* untuk kembali ke pertanyaan sebelumnya dan melakukan koreksi jawaban. Pada saat kembali ke pertanyaan tertentu, fakta setelah pertanyaan tersebut akan dibatalkan yang ditunjukkan dengan terhapusnya data pada data *history*.



Gambar 12. Tampilan Data History

User dapat terus-menerus melakukan *troubleshooting* sampai menemukan solusi yang dapat digunakan untuk mengatasi permasalahan komputernya. Pada sistem ini ada 2 jenis solusi yang disediakan, yaitu :

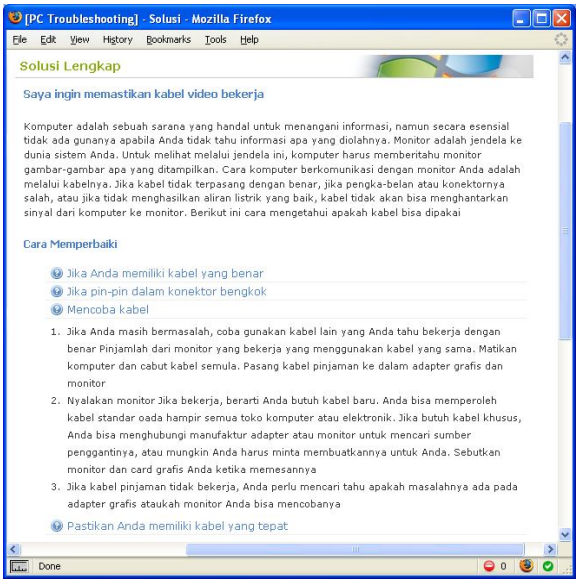
- **Solusi cepat/singkat**

Solusi cepat merupakan solusi yang sifatnya ringkas dan biasanya digunakan untuk mengatasi permasalahan yang tidak terlalu rumit.

Gambar 13. Tampilan Solusi cepat

- **Solusi lengkap**

Solusi lengkap merupakan solusi yang benar-benar memandu *user* tahap demi tahap dalam mengatasi permasalahan komputer. Pada solusi ini beberapa tahapannya juga desertai dengan gambar dan video.



Gambar 14. Tampilan solusi lengkap

4.3 Manajemen Basis Pengetahuan

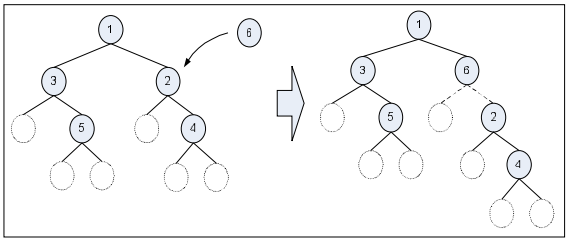
Penambahan aturan pada basis pengetahuan merupakan fasilitas yang sangat berperan dalam suatu sistem berbasis aturan, terutama dalam kaitannya dengan pengembangan sistem. Adanaya fasilitas ini akan sangat membantu dalam melakukan perbaikan aturan pada sistem ketika suatu saat terjadi perubahan aturan karena berkembangnya pengetahuan. Dengan demikian, sistem akan tetap *up to date.*

4.3.1 Manajemen Aturan Pada Basis Pengetahuan

Pada sistem pakar untuk *troubleshooting* komputer ini terdapat fasilitas untuk penambahan aturan yang dapat dilakukan dengan 3 cara, yaitu :

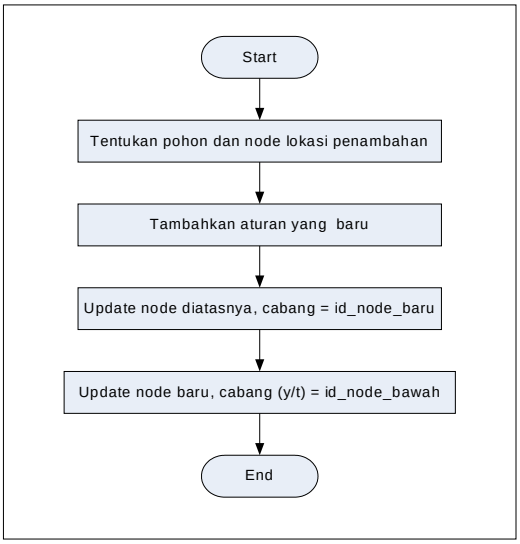
1. **Menambahkan aturan baru di tengah/menyisipkan pada pohon**.

Sebelum menambahkan aturan baru, terlebih dahulu harus ditentukan pohon serta *node* dimana aturan baru tersebut akan ditambahkan. Menambahkan aturan baru ditengah pohon akan berpengaruh pada kedua node, baik yang ada di atas maupun di bawahnya.



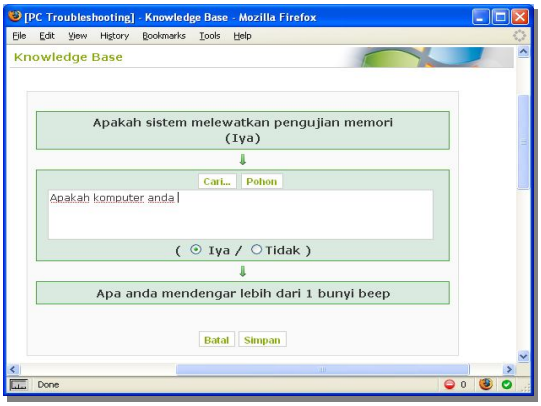
Gambar 15. Ilustrasi penyisipan aturan pada pohon

Aturan yang baru akan menggantikan posisi cabang yang lama dan cabang yang lama akan menjadi cabang dari *node* yang baru. Alur logika dalam penyisipan aturan baru pada pohon dapat digambarkan sebagai berikut.



Gambar 16. Alur dalam penyisipan aturan pada pohon

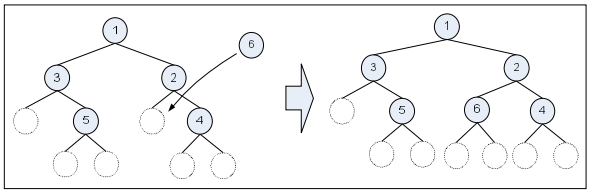
Implementasi penyisipan aturan baru pada fasilitas penambahan aturan yang terdapat pada sistem akan dilakukan seperti pada Gambar 17.



Gambar 17. Implementasi penyisipan aturan pada pohon

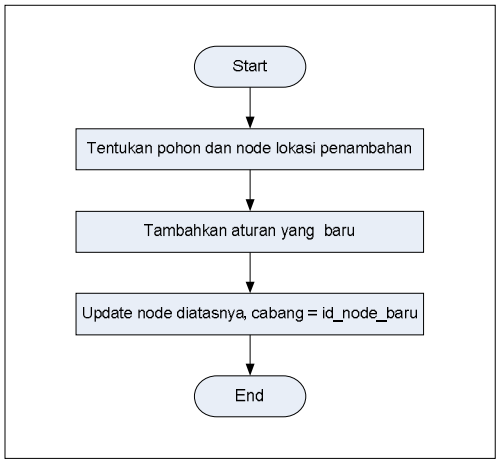
4.3.2 Menambahkan aturan baru di akhir pohon

Untuk menambahkan aturan baru diakhir pohon, *user* harus memilih *node* dengan cabang yang masih kosong. Menambahkan aturan baru diakhir pohon hanya akan mempengruhi *node* yang dipilih.



Gambar 18. Ilustrasi penambahan aturan pada akhir pohon

Aturan yang baru ditambahkan pada akhir pohon tersebut akan menjadi cabang dari *node* yang dipilih. Alur logika dalam penambahan aturan baru ini dapat digambarkan sebagai berikut.



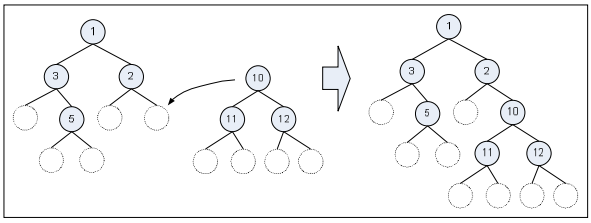
Gambar 19. Alur dalam penambahan aturan pada akhir pohon

Pada fasilitas penambahan aturan pada sistem, implementasi penambahan aturan pada akhir pohon akan sedikit berbeda dengan penyisipan di tengah pohon. Seperti pada gambar 20, setelah aturan yang baru tidak ada lagi pertanyaan lanjutan di bawahnya.

Gambar 20. Implementasi penambahan aturan pada akhir pohon

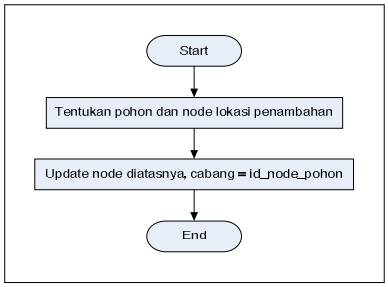
4.3.3 Menambahkan aturan dari pohon yang sudah ada

Sama seperti saat penambahan aturan pada akhir pohon, penambahan aturan dari pohon yang sudah ada juga hanya dapat dilakukan pada cabang yang masih kosong. Pada penambahan ini tidak ada aturan baru yang dibuat, yang terjadi hanya perubahan pada aturan yang sudah ada.



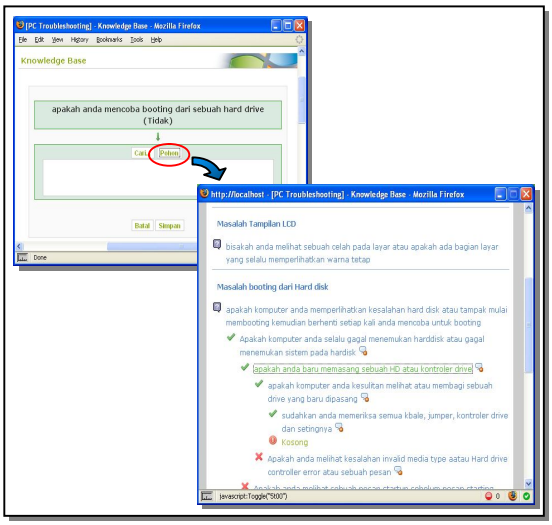
Gambar 21. Penambahan aturan dari pohon yang sudah ada

Penambahan aturan dari pohon yang sudah ada mengakibatkan cabang yang sebelumnya masih kosong akan mengadopsi semua aturan dari pohon yang ditambahkan. Berikut adalah alur logika pada penambahan aturan dari pohon yang sudah ada.



Gambar 22. Alur dalam penambahan aturan dari pohon yang sudah ada

Implementasi penambahan aturan ini dilakukan dengan memilih *node* dari pohon yang sudah ada seperti pada Gambar 23, sehingga pada proses ini tidak ada aturan baru yang dibuat.

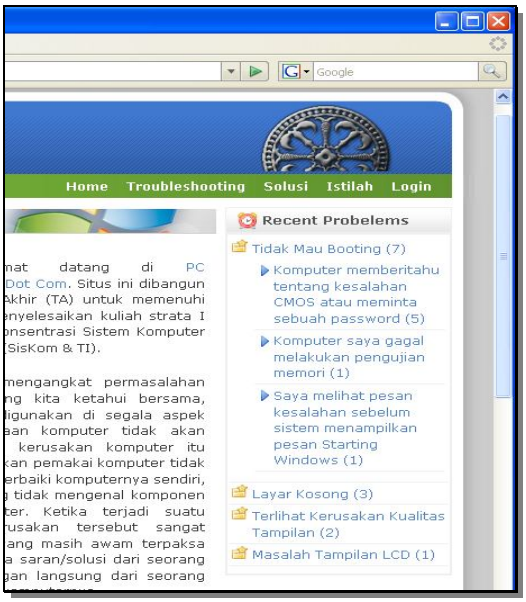


Gambar 23. Implementasi penambahan aturan dari pohon yang sudah ada pada sistem

4.4 Kelebihan dan Kekurangan Sistem

Secara umum Sistem pakar untuk *Troubleshooting* Komputer ini memiliki beberapa kelebihan, antara lain :

1. Permasalahan yang ditangani sistem dikelompokkan menjadi beberapa kategori, sehingga identifikasi masalah dapat dilakukan dengan cepat dan *user* tidak perlu melalui penelusuran yang terlalu panjang untuk menemukan sebuah solusi.
2. Fasilitas *history* yang ada pada sistem memungkinkan *user* untuk kembali ke pertanyaan manapun yang telah dijawab sebelumnya dengan cepat. Setelah kembali ke pertanyaan yang diinginkan *user* juga dapat langsung melakukan koreksi jawaban.
3. Untuk menambahkan permasalahan baru, *Knowledge Engineer* dapat melakukan pembentukan pohon dengan mengadopsi pohon yang sudah ada. Dengan demikian maka data pada *database* menjadi efisien karena tidak terjadi pengualangan data yang sama serta akan memudahkan *knowledge engineer* dalam menambahkan data pengetahuan.
4. Setiap permasalahan yang ditemukan pada saat proses *Troubleshooting* disimpan pada *database* dan dikelompokkan menurut gejala utamanya. Pada halaman *website* data ini ditampilkan sebagai *Recent Problems*, sehingga memungkinkan *user* dapat dengan cepat menemukan solusi atas kerusakan komputernya.



Gambar 24. Tampilan *Recent Problems*

5. Kebanyakan sistem pakar belum memiliki fasilitas untuk menambah pengetahuan, namun pada sistem untuk *troubleshooting* ini telah menyediakan fasilitas manajemen pengetahuan sehingga lingkup permasalahan yang ditangani dapat dikembangkan oleh *knowledge engineer* dengan mudah.
6. Basis pengetahuan yang tersimpan pada *database* dan adanya fasilitas manajemen pengetahuan akan memungkinkan sistem ini dikembangkan menjadi sistem untuk identifikasi permasalahan yang lain.

Disamping memiliki kelebihan seperti yang dipaparkan di atas, sistem pakar ini juga memiliki beberapa kekurangan, seperti :

1. Tidak adanya fasilitas penjelasan untuk setiap pertanyaan pada saat proses identifikasi masalah yang mengakibatkan user cenderung menjawab "tidak mengerti" ketika kurang paham terhadap maksud pertanyaan.
2. Format file yang dapat diupload pada penambahan detail solusi masih sangat terbatas, untuk file gambar hanya format "JPG" dan untuk video hanya "WMV"
3. Pembuatan aturan yang dilakukan dengan memakai pohon yang sudah ada akan membuat data basis pengetahuan pada *database* tetap efisien, namun hal ini juga memiliki kelemahan. Ketika dilakukan perubahan pada pohon yang dipakai bersama, maka perubahan juga akan terjadi pada pohon yang mengadopsi pohon tersebut.
4. Suatu solusi akan ditemukan jika fakta dari aturan yang terakhir telah ditentukan. Ketika proses identifikasi masalah, jika user tidak

menentukan fakta dari aturan tersebut (menjawab 'tidak mengerti'), maka sistem tidak akan bisa memberikan solusi.

5. SIMPULAN

1. Pengembangan sistem pakar untuk memperbaiki kecepatan dan kegagalan koneksi peralatan eksternal dilakukan dengan pembentukan *tree* paada basis data MySQl sehingga memiliki kedinamisan dalam manajemen pengetahuan/kepakaran
2. Pemilihan bahasa pemrograman web dengan PHP akan memberikan fleksibelitas dalam pengaksesan data
3. Pendekatan pengembangan user interface dilengkapi dengan fasilitas pengenalan istilah dan dokumen penunjang berbasis multimedia sehingga memberikan kemudahan pemakaian bagi pemakai awam

6. DAFTAR PUSTAKA

[1]. Arhami , M.,2005, Konsep Dasar Sistem Pakar, ANDI Yogyakarta, Yogyakarta.
[2]. Kadir, A. 2003. Pemrograman WEB Dinamis Menggunakan PHP. Yogyakarta : Andi
[3]. Kusumadewi, S. 2003. Artificial Intelligence (Teknik dan Aplikasinya). Yogyakarta : Graha Ilmu.
[4]. Natalia,D.A.,2006, Pembangunan Sistem Pakar pada Perangkat Mobile dengan WML dan PHP Untuk Penyakit Paru pada Anak, ITS Surabaya
[5]. Stone, D.M., dan Poor ,A. ,2001, Troubleshooting your PC, Elexmedia Computindo.
[6]. Turban, E; dan Jay E.A, 1995 Decision Support System and Intelligent System, six edition, Prentice Hall Internasional, Inc. New Jersey .
[7]. www.mspress.microsoft.com/troubleshooting
[7].www.swidodo.wordpress.com/tag/ uncategorized/artikel/membuat prototipe sistem pakar