dipilih *user*. Informasi ini merupakan titik tolak sistem pakar dalam melakukan proses pelacakannya.

Inference engine(motor inferensi) mulai melakukan pelacakan, mencocokan kaidah dalam pangkalan pengetahuan terhadap informasi yang ada di dalam pangkalan data. Bila setiap kaidah bisa merubah isi pangkalan data, maka status masalah akan terperbaiki. Fakta baru tersedia untuk digunakan dalam proses pengambilan keputusan. Selanjutnya, fungsi khusus seperti permintaan penambahan informasi dari user bisa dipicu.

Pangkalan data juga menyimpan daftar kaidah yang sudah diakui. Hal ini membantu proses *tracking*. Urutan kaidah bisa diberikan kemudian jika user memerlukan penjelasan tentang proses penalaran. Urutan kaidah bisa diberikan kemudian jika user memerlukan penjelasan tentang proses penalaran.

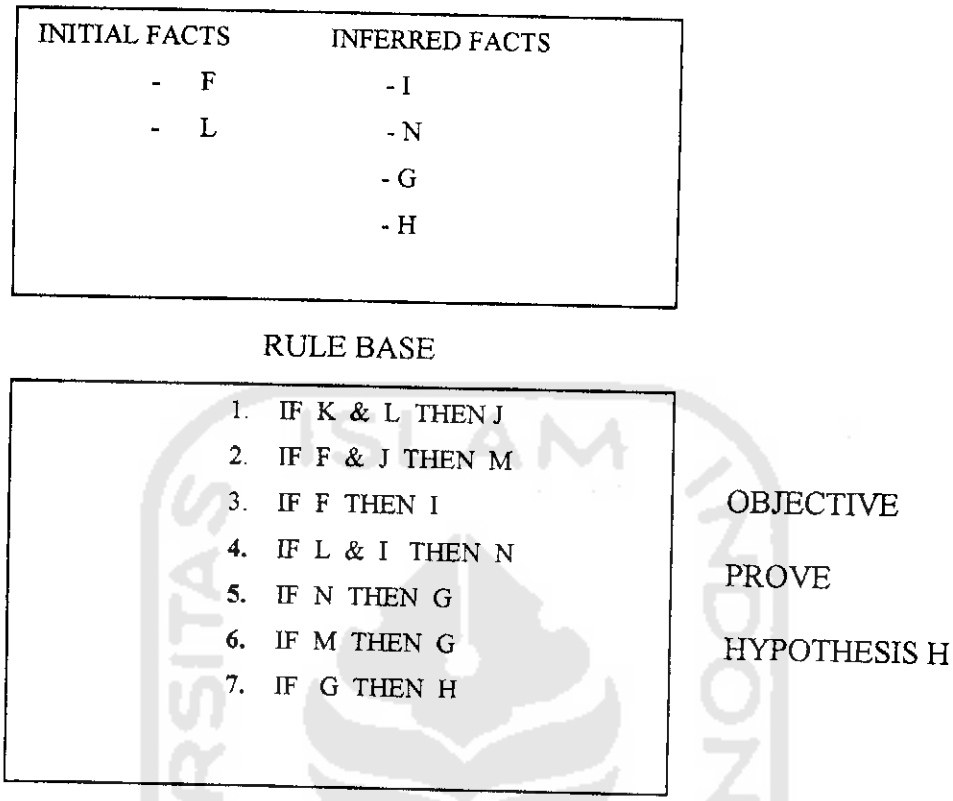
2.2.3.3 *Inference Engine*

Inference engine(motor inferensi) adalah *sofware* yang memerlukan alat operasi pelacakan dan penyocokkan pola. Kadang-kadang juga disebut penafsir kaidah, karena bekerjanya seperti *interpreter* bahasa komputer. Jika *interpreter* bahasa melihat baris kode ke dalam suatu program dan kemudian melakukan *spesifikasi* operasi, sedang *interpreter* kaidah menguji kaidah-kaidah dalam urutan tertentu untuk mencari yang sesuai dengan kondisi awal dan kondisi berjalan yang sudah dimasukkan ke dalam pangkalan data. Begitu kaidah cocok dengan kondisi yang ditemukan, maka kaidah *terstimulasi*, dan oleh karena itu mulai langkah selanjutnya ditentukan.

Karena kaidah terus *menstimulasi*, maka kaidah yang satu akan mengacu kepada kaidah lainnya dan membentuk lingkaran *inferensi*. Pada saat kaidah baru diuji, ia mengecek status solusi masalah yang sidah disimpan dalam pangkalan data. Hal ini memberi informasi tambahan kepada kaidah *interpreter* yang menyebabkan ia bisa me*modifikasi* urutan di dalam mana kaidah berikut terpilih. Proses ini terus berlangsung sampai ditemukannya solusi yang diinginkan.

Jelasnya *inference engine* merupakan bagian dari sistem pakar yang bertugas untuk menemukan solusi yang tepat dari banyaknya solusi yang ada. Proses yang dilakukan dalam *inference engine* ini adalah melakukan pengambilan keputusan terhadap konsultasi yang terjadi dan proses penalaran pada basis pengetahuan yang dimiliki. Penentuan pengambilan keputusan dan metode pelacakan sangat penting dalam rangka untuk menyelesaikan suatu permasalahan. Penentuan tersebut juga dilihat berdasarkan metode *representasi* pengetahuan yang digunakan dan kecocokan terhadap permasalahan utama (*problem domain*). Dalam proses penalaran pengetahuan dalam basis pengetahuan sistem pakar adalah dua bagian yaitu : secara *Backward Chaining* atau *Forward Chaining* . Keduanya bagian ini tergantung dari permasalahan yang dihadapi dan *efisiensi* proses penalaran pengetahuan yang ada dalam basis pengetahuan .

Forward Chaining atau data *driven search* atau *even driven search* adalah strategi pengambilan keputusan dimulai dari premis atau fakta-fakta kemudian apabila dari rule-rule tersebut ada yang cocok maka goal didapatkan dari konklusi *rule* tersebut. Contoh sederhana dibawah ini menunjukan pangkalan kaidah yang terdiri dari 7 buah [SUP91] :



Gambar 2.2 Contoh hipotesa sederhana sistem pohon menggunakan forward chaining

Backward chaining atau *goal driven* atau *directed reasoning* adalah strategi pengambilan keputusan atau kesimpulan dengan cara memulai dari bagian *konklusi* (dari belakang) kemudian kegiatan *premise* dinalar berdasarkan *konsultasi,* apabila *premise* tersebut sesuai atau terbukti maka goalnya *konklusi* tersebut. Contohnya berhubungan dengan gambar *forward chaining* di atas.

Misalkan jika K tidak ditemukan dalam pangkalan data , seperti gambar diatas , maka ia akan bertanya kepada user apakah tahu atau tidak. Bila tidak tahu, maka ia akan memasukannya. Kemudian sistem akan terus membuktikan H dengan menggunakan kaidah 1, 2 dan 7 dan tidak membuktikan 3, 4, 5 dan 7 . Kemampuan

bertanya, jika ia tidak mengetahuinya merupakan kelebihan dari sistem *backward chaining*.

Mekanisme penalaran *backward chaining* dapat digambarkan sebagai berikut :

1. Sistem diberi satu variabel goal sebagai tujuan utama dari pencarian
2. Untuk setiap *variabel goal* sistem akan mencari *rule* yang mempunyai *variabel konklusi* yang sam dengan *variabel goal* tersebut.
3. Dari setiap rule yang ditemukan ini sistem akan mengevaluasi semua *klausa*nya. Bila *variabel klausa* merupakan *variabel konklusi* maka *variabel klausa* ini akan menjadi *goal* baru. Tetapi bila *variabel klausa* ini bukan merupakan *variabel konklusi* maka sistem ini mengajukan pertanyaan kapada *user*.
4. Step 2 dan 3 diulang terus menerus selama *variabel goal* belum mempunyai nilai.

Untuk dapat memproses *knowledge base* dan menghasilkan sebuah *rekomendasi*, *backward chaining* membutuhkan bantuan berupa struktur data, antara lain :

1. Daftar Konklusi

Daftar *konklusi* merupakan suatu daftar *variabel konklusi* dari setiap *rule* pada *knowledge base* . Daftar ini terdiri dari *variabel konklusi* dan alamat *klausa* pertama. Daftar *konklusi* ini digunakan untuk mencari *rule* yang mengandung variabel konklusi tertentu. Daftar inisangat diperlukan karena pencarian pada backward chaining selalu dimulai dari konklusi menuju ke klausa.

2. Daftar Variabel

Daftar *variabel* merupakan daftar dari semua *variabel* yang terdapat pada *knowledge base* baik berupa *variabel konklusi* maupun *variabel klausa*. Daftar ini berfungsi untuk menunjukkan status *variabel* apakah sudah terisi belum dan juga untuk menyimpan nilai *variabel* beserta nilai faktor keyakinannya.

3. *Daftar Variabel Klausa*

Daftar *variabel klausa* memuat klausa dari tiap-tiap *rule*. Daftar ini terdiri dari *variabel* dan nilainya. Sebuah *konklusi* dapat diterima apabila seluruh *klausa* dari *rule* tersebut terpenuhi.

4. *Daftar Stack*

Daftar *stack* berguna untuk membantu mengingatkan alamat *rule* dari *konklusi* yang hendak dibuktikan dan alamat *klausa* yang terkhir diperoleh dan ini semua dengan sifat *stack* yaitu *lifo* yang masuk terakhir akan menjadi yang pertama.

2.2.4 Ciri – Ciri Sistem Pakar Yang Diinginkan

Expert sistem mempunyai ciri keunggulan, oleh karena itu ia akam menjadi sistem yang jauh berguna. Keunggulan yang dimaksudkan adalah fasilitas penjelasan, kemudahan modifikasi, bisa digunakan data, berbagai jenis komputer dan kemampuan *adaptif*. Keunggulan-keunggulan sistem pakar adalah :

1. Fasilitas Penjelasan

Sistem pakar yang baik adalah sistem pakar yang bisa memberikan informasi tentang kesimpulan yang diambil oleh komputer. Informasi yang diberikan hendaknya dengan memperlihatkan kaidah-kaidah yang dipergunakan dalam pengambilan keputusan dan urutan yang dilaksanakan.

2. Kemudahan *Modifikasi*

Integritas pangkalan pengetahuan tergantung kep*ada keakuratan dan up-to-date-nya pangkalan pengetahuan. Dalam* domain pengetahuan yang sering mengalami perubahan, adalah sangat penting untuk menyediakan sarana tertentu yang memudahkan dan mempercepat mengadakan perubahan.

2.2.5 User Interface

User interface merupakan media komunikasi antara *user* dengan mesin komputer melalui keyboard atau monitor . User interface ini memberikan dialog terhadap *user* serta menampilkan keputusan yang telah didapat oleh *inference engine.*

User interface yang dikembangkan harus disesuaikan dengan pemakai utama (*end user*) dari sistem pakar yang dikembangkan sehingga dapat tercipta komunikasi yang baik antara sistem dengan *user* . Dengan user *interface* mudah dan sederhana tentunya sangat membantu user dalam melakukan konsultasi dengan sistem pakar

2.2.6 Kaidah-Kaidah Produksi

Didalam suatu kaidah produksi sangat populer karena formatnya sangat *fleksibel.* Hanpir semua macam pengetahuan dapat ditulis dalam bentuk yang sesuai dengan formatnya kaidah *IF-THEN.* Kaidah semacam ini umumnya sanagt mudah ditulis, dan secara relatif mudah membentuk pangkalan pengetahuan yang *impresif* dengan cepat.

Pertama-tama yang harus selalu perlu diingat, bahwa kaidah di*format* ke dalam dua bagian. Pertama bagian *IF* yang menyatakan *premis*, kondisi atau a*ntecedent.* Biasanya disebut kaidah sebelah kiri. Kedua, bagian kaidah produksi

THEN yang menyatakan konklusi , seksi atau konsekuensi yang akan menggantikan kondisi sebelah kiri, bila ternyata sudah sesuai. Jika *premis* benar atau kondisinya cocok, maka bagian sebelah kanan juga benar. Kaidah ini disebut picu. Jika kaidah sebelah kanan bisa diimplementasikan, maka kaidah tersebut sudah dipicu. Beberapa kaidah yang diberikan dalam contoh dibawah ini menunjukkan beberapa bentuk pengetahuan yang diekspresikan kedalam bentuk format kaidah produksi :

IF Binatang itu hidup dalam air

AND Binatang itu bernafas dalam air

THEN Bimatang itu adalah ikan. CF 1.0

IF Pasien sering sakit kepala

AND Penglihatannya kabur

THEN Pasien harus memakai kaca mata. CF 0.8

Seperti anda lihat dari contoh tersebut diatas, setiap kaidah tersusun dari apa yang disebut *clause* . Clause adalah semacam kalimat yang terdiri dari *subjek*, kata kerja dan objek yang menyatakan beberapa fakta. Pada setiap *clause* terdapat satu *clause IF* dan satu clause *THEN* . Bagian kaidah *IF* mungkin berisi lebih dari satu *premis*, dan masing – masing berisi *clause*. Setiap kaidah yang mempunyai lebih dari satu *premis* disebut *compound clause* dan dihubungkan oleh kata penghubung *AND* atau *OR.*

CF adalah singkatan dari *certainty factor* (faktor kepastian) nomor 0 dan 1 menunjukkan tingkat kepercayaan *kepada kebenaran atau keabsahan suatu kesimpulan yang diberikan kepada* konklusi kaidah tersebut ditas.

Jenis pengetahuan yang bisa ditampilkan dengan kaidah-kaidah adalah sangat luas. Pengetahuan teori yang sudah ada dalam buku-buku biasanya bisa dialihkan kedalam format kaidah. Hampir disemua sistem yang dipakai dalam kehidupan sehari-hari, membutuhkan pengetahuan yang sudah diakumulasi oleh seorang pakar selama bertahun-tahun. Hal ini disebut pengetahuan *empiris*, yaitu jenis pengalaman *cut-and-error* yang hanya didapatkan dari keterlibatan langsung dalam pemecahan masalah secara praktis. Pengetahuan ini disebut pengetahuan *heuristik.*

Disamping ini juga jangan sampai lupa, bahwa setiap kaidah produksi menampilkan sekelumit pengetahuan individual. Hal ini biasanya dihubungkan dengan banyak kaidah . Kaidah tersebut dikaitkan secara bersama-sama untuk membentuk garis penalaran. Misalnya, kesimpulan dari satu kaidah mungkin bisa menjadi *premis* bagi yang lainnya. Ini adalah kumpulan atau jaringan kaidah yang membentuk pangkalan pengetahuan. Selama elemen dasar dari semua pangkalan pengetahuan merupakan kaidah produksi, maka anda akan sering mendengarkannya sebagai baris kaidah .

2.2.7 Alat Bantu Pengembangan Sistem Pakar

Software yang bisa digunakan dalam pembuatan sistem pakar adalah *prolog* atau *LISP.* Namun demikian *Pascal*, C, atau *basic* juga dapat dipakai dengan sejumlah keterbatasan yang ada didalamnya, terutama adalah kesulitan untuk pengembangan sistem pakar, yang biasanya disebut dengan *Expert System Shell.* Saat ini *ES-Shell* menjadi cara yang paling banyak dipakai untuk pengembangan sistem pakar skala kecil (*Prototype Expert System*) melalui *mikrokomputer*. Dapat disebut

sejumlah *ES-Shell* yang terdapat dipasaran *software*, seperti *VP -Expert*, *Exsys*, *M.I*, *ESIe*, Guru dan level 5 .

2.2.8 Langkah-Langkah Pembuatan Sistem Pakar

Mengetahui bagaimana sistem pakar bekerja dan bagaimana mengoperasikannya meruapakan satu hal. Hal lainnya adalah bagaimana cara mengambangkan atau membuat sistem pakar. Sebetulnya menggunakan sistem pakar bukanlah hal yang terlalu sulit, dan boleh juga dikatakan hal yang mudah. Yang sulit adalah bagaimana cara merancang dan membuat sistem pakar itu. Untuk membuat satu program sistem pakar, mulai dari coretan sampai selesai memerlukan banyak pemikiran, rancangan, pemrograman dan *debugging*. Memang sistem pakar merupakan progran komputer yang sangat rumit dibanding dengan program komputer *konvensional* lainnya.

Walupun demikian, tidak perlu kecil hati atau menyerah sebelum memulai. Sudah banyak program-program sistem pakar yang sudah berhasil dengan baik. Artinya bagimanapun sulitnya toh masih bisa juga dibuat dan dikembangkan.

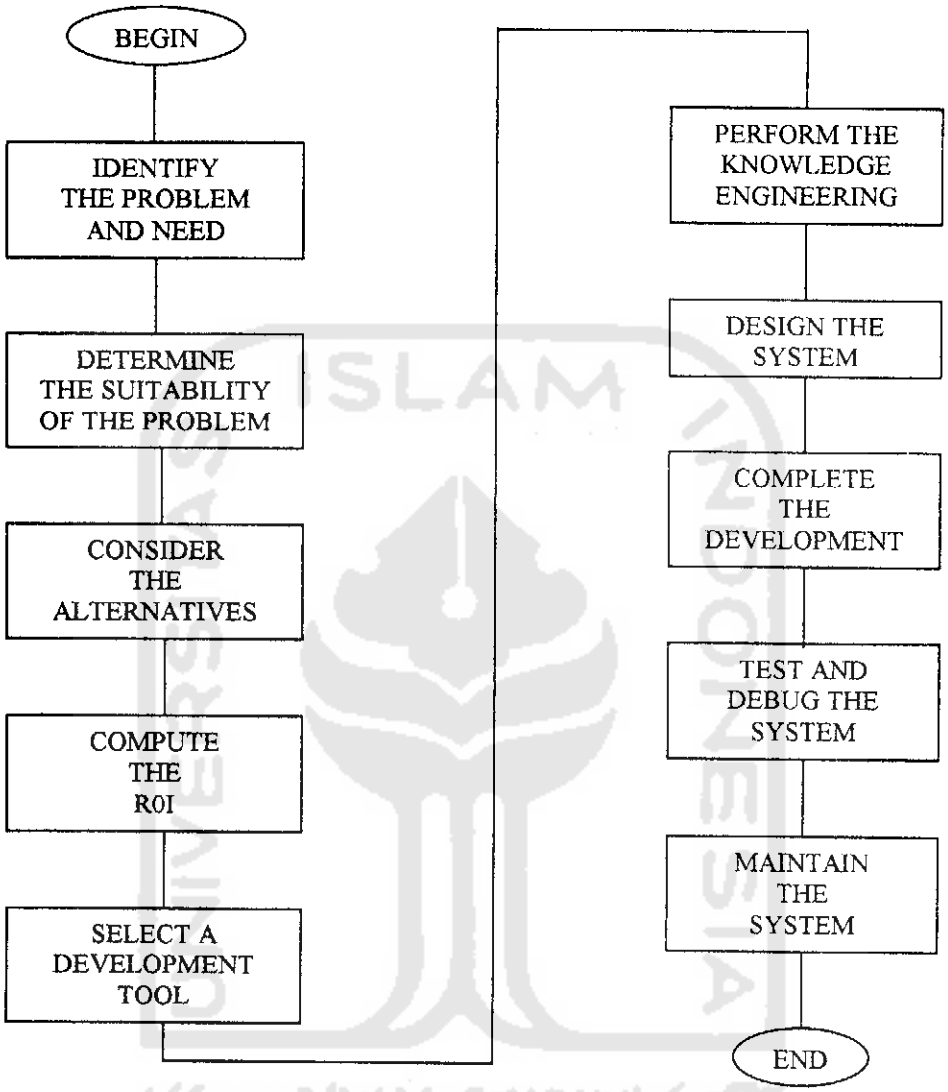
Walaupun dalam pembuatan sistem pakar langkah-langkahnya sudah dirumuskan dengan jelas, tapi harus selalu diingat bahwa ukuran, *scope* dan pokok soalnya bisa sangat beraneka ragam. Sedangkan langkah-langkahnya itu sendiri hanya merupakan garis besar. Satu-satunya cara belajar membuat atau mengembangkan sistem pakar adalah membuat sendiri sistem pakar yang benar-benar *aktual*. Dimulai dengan mengikuti gairs besar dasar, memilih topik yang sederhana, kemudian mencoba sendiri. Pasti anda akan berhasil. Pengalaman yang anda peroleh selama proses pembuatan sistem pakar yang kecil dan sederhana itu,

akan menjadi modal yang sangat berharga untuk pembuatan sistem pakar yang lebih besar dan lebih *ambisius* lagi.

Dalam hal ini yang selalu diingat yaitu pertama, proses pengembangan seperti digambarkan adalah agar anda menggunakan alat pemrograman yang sudah ada, yaitu paket *software* yang akan membantu anda mengembangkan sistem pakar dengan tidak perlu membuat program sendiri. Alat pemrograman yang sudah digunakan untuk membuat sistem pakar adalah *Prolog, Lisp*, atau bahasa *konvensional* lainnya seperti pascal dan bahasa C.

Kedua akan menjelaskan pengembangan sistem pakar untuk komputer pribadi. Proses pengembangan umum sudah bisa diterapkan pada setiap jenis dan ukuran komputer, tapi akan memfokuskan khusus pada komputer pribadi, karena sekarang ini sudah banyak tersedia alat-alat pengembangan yang sudah bisa digunakan.

Dalam pembuatan sistem pakar ada sepuluh langkah yang perlu diperhatikan. Gambar 2.3 menunjukkan proses pembuatan sistem pakar yang sederhana langkah demi langkah.



Gambar 2.3 Langkah-langkah dalam proses membuat sistem pakar

2.2.8.1 Mengindetifikasikan Masalah Dan Kebutuhan

Pada dasarnya sistem pakar pun sama yaitu suatu *solusi* yang menjawab masalah.Agar pembuatan sistem pakar bisa dibenarkan, maka harus ada satu masalah yang harus dipecahkan atau harus dicocokkan. Untuk alasan ini, maka langkah pertama yamng harus dilakukan adalah mengkaji situasi dan memutuskan dengan

pasti tentang masalah yang akan dikomputerisasi dan apakah dengan sistem pakar bisa lebih membantu atau tidak.

Dalam berusaha untuk memperoleh suatu hasil yang memuaskan , sering dihadapkan kepada *problema,* yaitu *problema* waktu, *produktivitas* , dan *problema* orang. *Problema* yang anda *identifikasi* harus benar-benar cocok untuk solusi sistem pakar.

2.2.8.2 Menentukan Problema Yang Cocok

Jika masalah sudah di*identifikasi* dengan jelas, kemudian anda mengkajinya lebih mendalam untuk mengetahui apakah tepat untuk sistem pakar atau tidak. Walaupun anda sendiri mungkin terpesona oleh konsep sistem pakar dan mencari jalan untuk menerapkannya pada pekerjaan anda, tapi harus selalu ingat bahwa hanya masalah tertentu saja yang bisa dipecahkan dengan sukses oleh sistem pakar.

Dengan menggunakan arahan umum seperti yang sudah digambarkan sekarang akan dicoba disajikan, anda akan sedikit mendapat kesulitan dalam menentukan apakah problema yang akan anda pilih itu cocok atau tidak untuk dijadikan calon. Faktor kunci untuk mencarinya adalah *aplikasi* pengetahuan pakar untuk bisa memecahkan masalah tersebut.

Untuk bisa mengambil keuntungan dari sistem pakar, anda harus mempunyai komputer atau setidak-tidaknya bisa menggunakan komputer lain untuk meng-akses sistem pakar tersebut. Tapi sekarang ini, soal komputer sudah tidak menjadi masalah, karena sudah banyak perusahaan atau perorangan yang menggunakan komputer. Oleh karena itu sistem pakar punya hak hidup. Suatu masalah lagi yang paling pokok

adalah pengetahuan yang dimiliki oleh pakar-pakar tidak akan berguna jika tidak bisa memperoleh dan memanfaatkannya dengan cepat dan mudah.

2.2.8.3 Mempertimbangkan Alternatif

Katakanlah anda sudah bisa mendapatlkan masalah yang anda anggap cocok untuk diterapkan dalam sistem pakar. Tapi walaupun demikian anda tidak boleh menganggap enteng atau menyamaratakan bahwa *solusi alternatif* itu akan memuaskan . Contohnya, seorang karyawan yang mengerjakan masalah itu harus yang sudah berpengalaman. Disamping itu juga semua karyawan yang terlibat dalam pekerjaan ini harus diberi informasi yang luas agar dia mampu mengerjakan hal-hal yang bersifat manual. Karena solusi *non* komputer bukan hanya akan merupakan *solusi* terbaik, tapi juga akan merupakan solusi yang lebih sederhana dan tidak terlalu mahal.

Pertimbangan *alternatif solusi* lainnya adalah *DBMS* mungkin lebih tepat untuk masalah yang akan anda kerjakan. Karena dengan *DBMS* cukup untuk menyimpan data-data dan informasi-informasi itu dan tinggal meng-aksenya bila diperlukan. Dalam hal *DBMS* ini, kemampuan menalar komputer tidak diperlukan . Akhirnya jangan mengabaikan solusi *software konvensional* . *Solusi algoritma* yang langsung mungkin lebih baik daripada *solusi simbolik AI*.

Sesudah mengetahui alternatif mana yang akan dipilih, maka baru anda bisa membuat keputusan final . Jika sistem pakar memang menjadi pilihan yang tepat , bisa mulai melangkah ke tahap berikutnya.

Pendeknya sebelum menetukan sistem pakar sebagai pilihan utama , sebaiknya dikaji dulu *alternatif-alternatif* lain yang lebih mudah, cepat dan sesuai dengan masalah yang ingin selesaikan.

2.2.8.4 Menghitung Pengembalian *Investasi*

Jika pilihan jatuh kepada sistem pakar maka langkah berikutnya adalah menentukan apakah sistem pakar lebih menguntungkan atau tidak. Harus menghitung pengembalian *investasi* dengan jalan menganalisis biaya dan kemungkinan keuntungan. Hal ini akan membantu dalam *investasi* pembuatan sistem pakar dan menentukan apakah biaya yang dikeluarkan itu akan sesuai dengan hasil yang akan dicapai.

Mengembangkan sistem pakar bukanlah pekerjaan yang sederhana. Ia akan memerlukan biaya yang sangat besar bukan hanya untuk pembelian *software* saja, tapi juga waktu yang diperlukan. *Personil* yang akan terlibat dalam pekerjaan itu dan bahkan untuk sistem pakar yang paling sederhanapun diperlukan waktu berbulan-bulan dan biayanya ribuan dolar. Dengan waktu dan biaya yang besar hasilnya akan memuaskan?

Menghitung pengembalian *investasi* sangat diperlukan dalam sistem pakar karena dengan dibuatnya sistem pakar apakah bisa lebih menghemat atau tidak? jika *asumsi* kerugian yang diderita cukup besar pada waktu pembuatan sistem pakar, maka sebaiknya proyek pembuatan sistem pakar itu dipikir ulang agar bisa menghemat untuk jangka waktu tertentu.

Memang tidak mungkin untuk menghitung penghematan, tetapi masih ada hal lain yang bisa menenggang pengembangan. Hal ini akan diperoleh dari keuntungan penggunaan sistem pakar yang bisa melebihi dari pada biaya pembuatannya.

2.2.8.5 Memilih Alat Pengembangan

Alat pengembangan sistem pakar adalah paket *software* yang memungkinkan bisa memasukkan pengetahuan pakar ke dalam komputer tanpa harus membuat suatu program terlebih dahulu. Hampir semua alat pengembangan sistem pakar menggunakan pangkalan kaidah. Beberapa diantaranya menggunakan *implementasi frame* dan jaringan semantik, tapi bisa lebih mahal dan hanya bisa dioperasikan dalam komputer besar.

Pekerjaan berikut adalah meng*identifikas*ikan alat-alat yang sudah ada dan memilih salah satu diantaranya yang sesuai dengan apa yang akan diperlukan.

2.2.8.6 Merekayasa Pengetahuan

Akhirnya saatnya mulai mengerjakan karya kreatif nyata. Pengembangan sistem pakar dimulai dengan merekayasa pengetahuan, yaitu bagaiman cara memperoleh pengetahuan. Seperti diketahui pengetahuan bisa diperoleh dengan berbagai cara, yaitu melalui artilel-artikel ilmiah, buku-buku yang bisa diperoleh dengan mudah dan cepat. Pengetahuan aktual bisa diperoleh dari *individu* atau seseorang yang memang ahli dibidangnya. Waupun bisa memperoleh dari buku-buku, tapi toh masih tetap membutuhkan satu atau dua orang ahli yang khusus menekuni pekerjaan tersebut.

Format atau bentuk penngetahuan akan menuntun dan mengarahkan dalam memilih skema penampilan pengetahuan yang diperlukan. Jika pengetahuan itu merupakan pengetahuan yang luar biasa, maka bisa memastikan menggunakan *representasi* pengetahuan dalam bentuk kaidah produksi. Untuk itu selama tahap rekayasa pengetahuan , harus berusaha menyempurnakan banyak kaidah yang paling sesuai. Dengan demikian akan punya banyak pilihan alat pengembangan yang paling tepat.

2.2.8.7 Merancang Sistem

Dengan menggunakan pengetahuan yang sudah diperoleh dan alatnya sudah pilih, sekarang sudah bisa mulai merancang sistem pakar. Pertama membuat garis-garis besarnya, hirarki bagan alur, matrix, tabel keputusan atau format lain yang akan membantu dalam mengorganisasi dan memahami pengetahuan itu. Dengan menggunakan bantuan ini mulai mengkonversi pengetahuan dalam bentuk kaidah *IF-THEN* . Sebaiknya diikuti prosedur tertentu sesuai denngan yang disarankan oleh *software* yang digunakan. Bila sudah selesai baru mulai menggunakan alat untuk membuat *prototype* bagian sistem. Kemudian menerjemahkan bagian pengetahuan kedalam kaidah dan menguji bagian yang sudah dibuat baru. Hal ini dimaksudkan agar untuk menguji konsep sebelum melanjutkan pembuatan seluruh program.

2.2.8.8 Melengkapi Pengembangan

Jika sistem yang sudah dibuat sudah selesai dan ternyata sudah bisa berjalan sebagai mestinya seperti yang diharapkan, barulah melangkah pada peluasan *prototype* ke dalam sistem yang final.

Cara terbaik untuk mengerjakan hal ini ialah dengan jalan meluaskan bagian demi bagian. Secara khusus, pengetahuan itu akan dibagi kedalam " potongan " yang *logis*, masing-masing dengan blok kaidah. Setiap bagian diuji apakah sudah bisa berjalan sesuai dengan aslinya. Kerjakan kaidah demi kaidah. Dengan cara ini akan memperoleh kemajuan , dan akan berakhir pada suatu sistem yang bisa berjalan tanpa banyak kesalahan.

2.2.8.9 Menguji Dan Mencari Kesalahan Sistem

Sesudah sistem pakar dikembangkan, perlu menyisihkan waktu untuk menguji dan mencari kesalahan (*debugging*). Tidak pernah ada sistem pakar yang begitu dibuat begitu berhasil dengan sempurna, oleh karena itu diperlukan pekerjaan tambahan untuk menyempurnakannya.

Bawalah sistem pakar ke lapangan dan lakukan percobaan dengan *user* yang menginginkannya. *User* akan menunjukkan bagian mana yang harus dirobah, atau dikoreksi atau bila perlu dikurangi sesuai dengan keinginannya.

2.2.8.10 Memelihara Sistem

Beberapa data yang dibuat bersifat *statis*. Oleh karena itu sistem pakar harus selalu dipelihara dan dikembangkan sesuai dengan perkembangan pengetahuan itu sendiri. Misalnya diperbaharui pengetahuan, mengganti pengetahuan yang sudah ketinggalan dan meluweskan sistem agar bisa lebih baik lagi dalam menyelesaikan masalahnya.

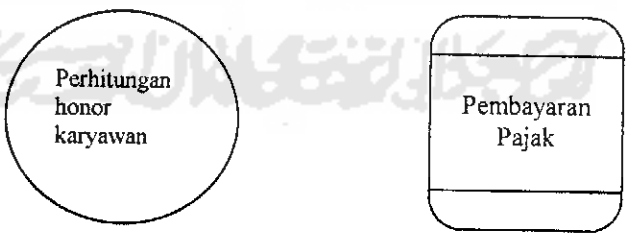
Bila sistem yang baru dibuat tidak selalu diperihara dengan baik, dalam arti selalu disesuaikan dengan perkembangan ilmu pengetahuan itu sendiri, maka sistem itu akan cepat menjadai tidak berguna dan investasi akan sia-sia.

2.3 Data Flow Diagram (DFD)

Model ini menggambarkan sistem sebagai jaringan kerja antar fungsi yang berhubungan satu sama lain dengan aliran dan penyimpanan data. Sebagai perangkat *analisis*, model ini hanya mampu memodelkan sistem dari satu sudut pandang yaitu sudut pandang fungsi. Komponen dari model ini adalah :

a. Proses

Komponen pertama dalam model ini dinamakan proses. Proses menunjukkan *transformasi* dari masukan menjadi keluaran, dalam hal ini sejumlah masukan dapat menjadi satu keluaran atau sebaliknya. Proses direpresentasikan dalam bentuk lingkaran atau bujursangkar dengan sudut melengkung. Bentuk proses dapat dilihat pada gambar 2.4 [POH97]



Gambar 2.4 Simbol Proses

b. Aliran

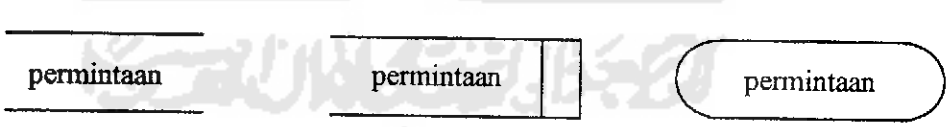
Komponen ini direpresentasikan dengan menggunakan panah yang menuju ke atau dari proses. Aliran digunakan untuk menggambarkan gerakan paket data

atau informasi dari satu bagian ke bagain lain dari sistem dimana penyimpanan mewakili lokasi penyimpanan data. Aliran yang digambarkan sebagai panah dengan dua ujung menggambarkan terjadinya dialog. Aliran dapat juga menyebar atau menyatu. Bentuk dari aliran dapat dilihat pada gambar 2.5 [POH97]

Gambar 2.5 Simbol Aliran

c. Penyimpanan

Komponen ini digunakan untuk memodelkan kumpulan data atau paket data. Notasi yang digunakan adalah garis sejajar, segiempat dengan sudut lengkung atau persegi panjang. Bentuk dari penyimpanan dapat dilihat pada gambar 2.6 [POH97]



Gambar 2.6 Simbol Penyimpanan

d. Terminator

Komponen ini dalam model direpresentasikan menggunakan persegi panjang, yang mewakili entiti luar dimana sistem berkomunikasi. Notasi ini melambangkan dari orang atau kelompok orang misalnya organisasi di luar

sistem. Pada sejumlah kasus dapat juga merupakan sistem lain. Bentuk dari terminator dapat dilihat pada gambar 2.7 [POH97]

Gambar 2.7 Simbol Terminator