

# BAB II

## SISTEM BERBASIS PENGETAHUAN

### 2.1 Sistem Berbasis Pengetahuan

Menurut [JAC99], sistem pakar adalah sistem yang secara umum bekerja dengan cara mengaplikasikan *rule of thumb* pada suatu pengetahuan yang direpresentasikan secara formal, tanpa menggunakan metode-metode algoritmik/statistik. Sistem pakar bertujuan untuk memecahkan masalah pada ranah tertentu. Sedangkan sistem berbasis pengetahuan lebih umum dari sistem pakar [PUP93], yaitu sistem yang bekerja pada pengetahuan yang direpresentasikan secara formal.

Karakteristik sistem pakar antara lain:

1. Mencoba mensimulasikan nalar manusia. Sistem pakar berusaha mengikuti langkah-langkah yang diambil oleh pakar dalam menyelesaikan masalah.
2. Menyelesaikan masalah dengan heuristik. Aturan-aturan pada domain persoalan tersebut biasanya berupa *rule of thumb* yang didapat dari pengalaman yang luas. Contohnya, seseorang menginginkan informasi perkiraan ongkos membangun rumah dari kontraktor dalam sekejap. Sang kontraktor untuk melakukan perkiraan secara detil perlu: memperkirakan biaya material, memperkirakan biaya buruh, memperhitungkan biaya darurat, serta menentukan untung yang layak. Melakukan itu semua memerlukan waktu yang banyak, akan tetapi dengan pendekatan heuristik, perkiraan biaya dapat jauh lebih cepat dicari: mengambil data rumah sejenis dan seukuran yang baru dikerjakan, normalisasi berdasarkan ukuran persisnya, kemudian penyesuaian untuk hal-hal yang sangat mempengaruhi biaya (jumlah kamar mandi, *finishing* yang mewah, dll).
3. Inferensi dilakukan pada representasi pengetahuan. Bagian yang menyimpan pengetahuan dipisah dari bagian yang menangani inferensi.

#### 2.1.1 Arsitektur Sistem Berbasis Pengetahuan

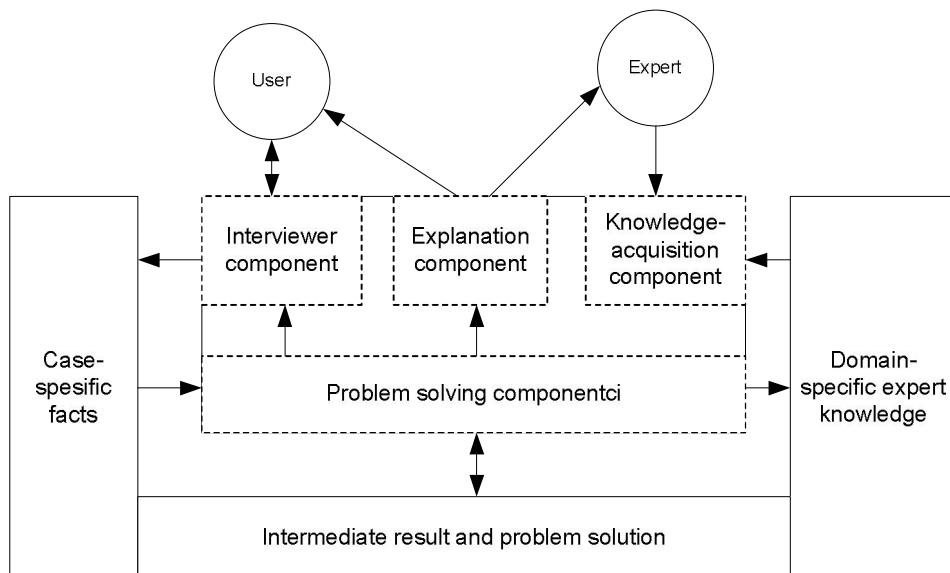
Secara garis besar, arsitektur SBP dibagi menjadi dua bagian utama, yaitu basis pengetahuan, dan mesin inferensi. Basis pengetahuan terbagi tiga (Gambar II-1), yaitu:

- Pengetahuan ahli yang spesifik pada ranah,
- Data kasus, berupa data kasus permasalahan yang ingin dicari solusinya

- Kumpulan data sementara yang didapat dari penarikan kesimpulan, disimpan pada *working memory*, dan salah satunya dapat merupakan solusi permasalahan.

Sedangkan mesin inferensi menangani strategi pemecahan masalah.

Selain basis pengetahuan dan mesin inferensi, terdapat komponen pendukung berupa antarmuka dengan pengguna. Antarmuka dapat dibagi-bagi lagi menjadi komponen untuk *query*, komponen untuk penjelasan (*explanation*), dan komponen untuk mengakuisi pengetahuan tambahan.



**Gambar II-1** Arsitektur SBP [PUP93]

### 2.1.2 Mesin Inferensi

Mesin inferensi adalah bagian sistem yang bertugas melakukan penarikan-penarikan kesimpulan dari data kasus dan data dalam basis pengetahuan. Pada sistem berbasis *rule*, siklus untuk mengaktifkan *rule* adalah sebagai berikut:

1. Cocokkan pola pada *rule* dengan daftar fakta
2. Jika lebih dari satu *rule* dapat aktif, maka dipilih satu (dengan strategi resolusi konflik)
3. Aktifkan *rule* (ada kemungkinan menambah/menghapus fakta) dan kembali ke (1).

Jika dua atau lebih *rule* dapat diaktifkan, maka strategi resolusi konflik menentukan cara pemilihan *rule*. Karena Tugas Akhir ini menggunakan CLIPS, maka di bawah ini dibahas strategi resolusi konflik yang digunakan oleh CLIPS [JAC99]:

1. *Depth strategy*, yaitu untuk *rules* dengan nilai *salience* yang sama, rule yang menggunakan data lebih baru akan didahulukan. Ini adalah strategi *default* pada CLIPS 6.0. Misalkan fact-a mengaktivasi rule-1 dan rule-2, fact-b mengaktivasi rule-3 dan rule-4. Jika fact-a di-*assert* sebelum fact-b, maka rule-1 dan rule-2 didahulukan daripada rule-3 dan rule-4.
2. *Breadth strategy*, yaitu untuk *rules* dengan nilai *salience* yang sama, rule yang menggunakan data lebih baru akan dikebelakangkan. Misalkan fact-a mengaktivasi rule-1 dan rule-2, fact-b mengaktivasi rule-3 dan rule-4. Jika fact-a di-*assert* sebelum fact-b, maka rule-3 dan rule-4 didahulukan daripada rule-1 dan rule-2.
3. *Specificity strategy*, yaitu *rule* yang memiliki *specificity* lebih sedikit akan didahulukan. *Specificity* adalah banyaknya perbandingan yang harus dilakukan pada bagian. Fungsi boolean **and**, **or**, dan **not** tidak menambah *specificity*. Contoh, rule berikut ini memiliki nilai *specificity* 5.

(defrule contoh

(item ?x ?y ?z) (test (and (p ?x) (> ?x (+ 10 ?y)) (< ?x 100)))

=>)

4. *Complexity strategy*, yaitu kebalikan *simplicity strategy*, rule yang memiliki *specificity* lebih banyak didahulukan
5. *LEX strategy*. Pertama-tama *rule* yang sudah pernah diaktifkan dipinggirkan. Dari sisanya dipilih berdasarkan baru-tidaknya data yang akan dipakai oleh prakondisi *rule* tersebut. Contoh, diketahui daftar rule di dalam agenda sebagai berikut:

rule-1: f-1, f-2, f-3

rule-2: f-3, f-1

rule-3: f-2, f-1

rule-4: f-1, f-4

dengan nilai indeks fakta yang lebih tinggi menandakan fakta yang lebih baru.

Dengan *LEX strategy*, maka urutan pengaktifan *rule* adalah:

rule-4: f-4, f-1

rule-1: f-3, f-2, f-1

rule-2: f-3, f-1

rule-3: f-2, f-1

6. *MEA strategy*. *Rule* yang sudah pernah diaktifkan dipinggirkan. Kemudian dari sisanya dipilih berdasarkan baru-tidaknya data yang akan dipakai prokondisi pertama *rule* tersebut. Misalkan diketahui daftar *rule* sama seperti contoh *LEX strategy* di atas. Dengan *MEA strategy*, maka urutan pengaktifan *rule* adalah:

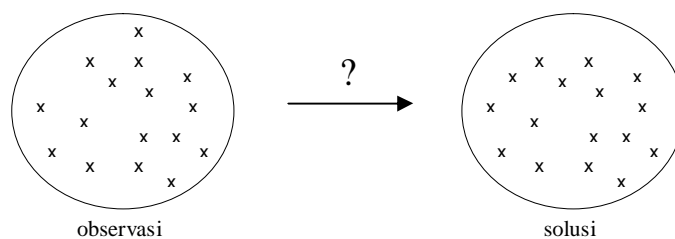
rule-2: f-3, f-1  
 rule-3: f-2, f-1  
 rule-4: f-1, f-4  
 rule-1: f-1, f-2, f-3

## 2.2 Metode Penyelesaian Masalah untuk Klasifikasi

Ada tiga tipe metode penyelesaian masalah dalam sistem berbasis pengetahuan [PUP93], yaitu:

- *klasifikasi*, yaitu solusi dipilih dari beberapa alternatif yang diberikan,
- *konstruksi*, yaitu solusi dibangun dari komponen-komponen primitif yang diberikan, dan
- *simulasi*, yaitu menentukan bagaimana suatu model sistem bereaksi terhadap masukan tertentu.

Pada sub-bab ini akan dijelaskan mengenai metode penyelesaian yang pertama, yaitu klasifikasi. Klasifikasi (digambarkan dengan Gambar II-2) dapat dianggap sebagai pencocokan antara kombinasi yang mungkin dari observasi ke solusi.



**Gambar II-2 Klasifikasi: pemetaan dari observasi ke solusi**

Karakteristik problem klasifikasi yaitu:

- Ranah terdiri dari dua himpunan, himpunan observasi dan himpunan solusi yang mungkin, dengan hubungan antara kedua himpunan ini yang rumit atau belum jelas.
- Suatu persoalan didefinisikan dengan suatu himpunan bagian dari observasi.
- Hasil adalah menentukan satu atau lebih penyelesaian dari masalah.

Untuk menyelesaikan problem klasifikasi, cara yang paling sederhana adalah menentukan hubungan satu-satu antara observasi dan solusi (disebut juga klasifikasi sederhana). Akan tetapi hal ini sulit dilakukan. Beberapa hambatan yang mungkin terjadi antara lain:

1. Pengetahuan yang tidak pasti.
2. Observasi yang kurang dapat diandalkan. Observasi macam ini terbagi lagi menjadi tiga macam, yaitu (a) observasi yang tidak pasti, (2) observasi yang subjektif, dan (3) observasi yang salah.
3. Observasi yang tergantung waktu, yaitu hasil observasi yang mungkin berubah kalau waktu pengamatan berubah.
4. Observasi tidak lengkap
5. Adanya abstraksi dengan menggunakan langkah perantara untuk mendapatkan solusi dari observasi.
6. Parameterisasi observasi maupun solusi
7. Solusi yang lebih dari satu.
8. Penggabungan solusi (jika lebih dari satu)

Klasifikasi sendiri dapat dibagi berdasarkan tipe pengetahuan yang digunakan.

#### 1. Klasifikasi sederhana

Klasifikasi sederhana dipakai jika pengetahuan hubungan antara observasi dan solusi sudah pasti. Implementasinya dapat menggunakan *decision tree* atau *decision table*.

#### 2. Klasifikasi heuristik

Jika solusi dapat diperkirakan atau dievaluasi dari observasi, maka yang umum digunakan adalah *klasifikasi heuristik*. Pengetahuan ini tidak pasti dan biasanya didapat dari pakar, sehingga hubungan antara observasi dan solusi memiliki

derajat atau nilai kepastian. Solusi yang dipilih adalah solusi yang derajat kepastiannya paling tinggi.

### 3. Klasifikasi *set-covering*

Pengetahuan yang dipunyai adalah suatu penyebab (solusi) akan menimbulkan efek tertentu (observasi). Jadi ini adalah hubungan dari solusi ke observasi, kebalikan dengan heuristik. Solusi yang dipilih adalah yang memuat observasi yang paling lengkap dan paling cocok.

### 4. Klasifikasi fungsional

Klasifikasi fungsional digunakan untuk *fault-finding system*, karena bisa membedakan antara fungsi normal dan tidak normal dari suatu sistem. Model fungsional menggambarkan sistem yang berjalan normal sesuai fungsinya, dan tidak menimbulkan gejala kesalahan apa-apa.

### 5. Klasifikasi statistik

Klasifikasi statistik mirip dengan klasifikasi heuristik. Perbedaannya yaitu, pada klasifikasi heuristik, hubungan observasi dan pengamatan didapat dari pakar, sedangkan pada klasifikasi statistik, hubungan ini didapat dengan data kasus dan dihitung dengan metode statistik.

### 6. Klasifikasi berbasis kasus

Pengetahuan yang disimpan adalah data-data kasus yang menghubungkan antara observasi dan solusi. Jika ada kasus baru, maka solusi dicari dari kasus yang paling mirip dengan yang sudah disimpan. Jika ada kesamaan, maka solusi diberikan. Oleh karena itu, klasifikasi ini memerlukan suatu cara perhitungan untuk menilai kesamaan.

## 2.3 Tahapan pembuatan SBP

Pembuatan sistem berbasis pengetahuan secara berurutan (Gambar II-3) adalah:

### 1. Analisis kelayakan.

Untuk tugas akhir ini, analisis kelayakan dilakukan oleh penulis berdasarkan *checklist* dari [SLA88] yang meliputi analisis kelayakan pengguna dan manajemen, analisis tugas (*task*), serta analisis pakar. Butir-butir yang perlu diperhitungkan ada di lampiran A, beserta penilaian untuk Tugas Akhir ini. Suatu proyek diperkirakan positif jika mendapat nilai 9,3 dan negatif jika kurang dari 7. Hasil analisis dapat disampaikan pada subbab 4.2.

## 2. Spesifikasi sistem

Spesifikasi yang dibuat meliputi latar belakang sistem, fungsi-fungsi sistem, dan batasan-batasan.

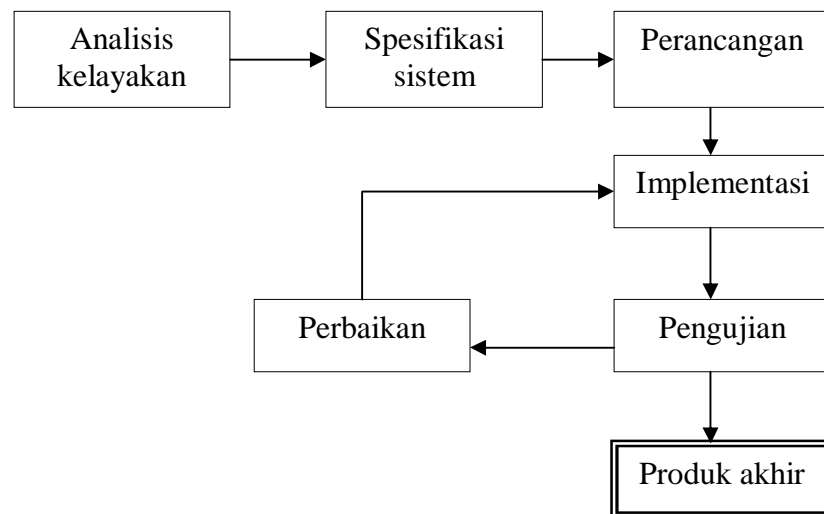
## 3. Perancangan

Perancangan meliputi rekayasa pengetahuan, pemilihan tool untuk implementasi, serta menentukan tim pengembang.

## 4. Implementasi, sesuai dengan rancangan

## 5. Pengujian

## 6. Perbaikan, jika ada kembali ke tahap 4.



**Gambar II-3 Tahap pembuatan SBP**

## 2.4 Rekayasa Pengetahuan

Salah satu tahap dalam pembuatan SBP adalah rekayasa pengetahuan. Rekayasa pengetahuan adalah akuisisi pengetahuan dalam suatu domain dari satu atau lebih sumber non-elektronik untuk dikonversikan ke dalam suatu representasi yang dapat digunakan oleh komputer. Komputer digunakan untuk memecahkan masalah yang umumnya hanya dapat dipecahkan oleh pakar domain tersebut. Dalam proses ini dilakukan akuisisi pengetahuan, yaitu mengekstraksi pengetahuan domain (*knowledge elicitation*) dan merepresentasikannya ke dalam bentuk formal.

Pada saat melakukan *knowledge elicitation*, dilakukan beberapa tahap secara berulang. Pertama-tama mengumpulkan pengetahuan (bisa dari pakar). Dari sana dapat ditentukan konsep utama domain permasalahan, dan menentukan hubungan-hubungan

antara berbagai konsep dalam domain. Lalu menentukan bagaimana pengetahuan direpresentasikan dalam sistem berbasis pengetahuan. Selanjutnya, menentukan pengetahuan apa yang perlu dikumpulkan selanjutnya. Sedangkan tahap-tahap untuk merepresentasikan pengetahuan yaitu:

1. Identifikasi karakteristik masalah
2. Konseptualisasi, yaitu menentukan konsep-konsep yang digunakan untuk representasi pengetahuan
3. Formalisasi, yaitu mendesain struktur organisasi pengetahuan
4. Implementasi, yaitu formulasi pengetahuan
5. Validasi pengetahuan

### **2.3.1 Akuisisi Pengetahuan**

Ada dua teknik untuk akuisisi pengetahuan, yaitu secara manual dan secara otomatis. Secara manual, dilakukan wawancara, observasi, atau secara intuisi. Sedangkan secara otomatis, digunakan tools untuk memfasilitasi akuisisi. Karena Tugas Akhir ini menggunakan cara manual, maka di sini dibahas mengenai wawancara, observasi, dan intuisi.

#### **1. Wawancara**

Wawancara dilakukan oleh pembuat sistem, pakar, dan pengguna akhir. Sebaiknya sebelum wawancara dilakukan wawancara pendahuluan yang bertujuan untuk memberitahukan tujuan SBP dan sosialisasi proyek kepada pakar, serta membangun hubungan yang baik antara pakar dan perekayasa pengetahuan.

Selanjutnya wawancara dilakukan untuk mengakuisisi pengetahuan. Yang diperlukan adalah diketahuinya tujuan wawancara, persiapan teknis wawancara (jadwal, alat perekam, dll), materi yang relevan dari pakar (jika ada), serta persiapan konten. Wawancara dapat dilakukan secara terstruktur atau tidak terstruktur.

#### **2. Observasi**

Pada cara observasi, pakar bekerja ketika memecahkan masalah, sedangkan perekayasa pengetahuan mengamati. Alternatif lain yaitu pakar memecahkan masalah, dan pengamat dapat berdiskusi atau menanyakan sesuatu pada pakar.

#### **3. Intuitif**

Pada cara ini, perekayasa pengetahuan dan pakar bertukar peran. Perekayasa memecahkan masalah, sedangkan pakar mengobservasi dan bertanya.

### 2.3.2 Representasi Pengetahuan

Pengetahuan yang diakuisi direpresentasikan dengan suatu representasi formal. Hal ini dilakukan agar pengetahuan bisa dimanipulasi dengan komputer sesuai dengan kebutuhan domain permasalahan. Representasi pengetahuan yang bisa dipakai untuk sistem berbasis pengetahuan dan sistem pakar adalah *rule system* dan *frames/object system*. Berikut ini dijelaskan masing-masing dari keduanya.

#### 1. Rule System

Sebuah rule terdiri dari bagian prekondisi dan bagian aksi. Biasanya rule ditulis dalam bentuk implikasi (*if-then*).

*if <prekondisi> then <aksi>*

Bagian prekondisi adalah pernyataan-pernyataan yang diperiksa ke basis pengetahuan. Jika bernilai benar, akan ditindaklanjuti dengan bagian aksi. Bagian aksi dapat berupa implikasi yang menambahkan pengetahuan baru, atau dapat berupa instruksi/langkah-langkah intruksi yang harus dijalankan.

Representasi formal untuk bagian prekondisi dan aksi bergantung pada kebutuhan, apakah cukup yang sederhana atau yang lebih ekspresif. Hal ini berpengaruh pada efisiensi *rule interpreter*. Evaluasi prekondisi yang paling sederhana adalah dengan *look up* ke basis pengetahuan disusul dengan relasi logika *and*, *or*, dan *not*. Yang lebih rumit adalah relasi yang memerlukan perbandingan harga, kalkulasi, urutan waktu, dan unifikasi.

Unifikasi menimbulkan masalah bagaimana mencocokkan sepasang variabel. Jika unifikasi tidak efisien jika setiap kali mencocokkan cara satu persatu. Algoritma Rete digunakan untuk mengatasi masalah ini.

Sistem berbasis *rule* terdiri dari data fakta yang benar, daftar *rule* untuk inferensi fakta-fakta baru, dan *rule interpreter* untuk mengontrol proses inferensi. Ada dua macam cara untuk memproses inferensi rule: *forward chaining* dan *backward chaining*. Pada *forward chaining*, rule-rule yang prekondisinya dipenuhi akan dibangkitkan sehingga dapat mengubah daftar fakta (menambah atau menghapus). Pembangkitan dilakukan terus-menerus sampai tidak ada lagi rule yang bisa aktif atau sampai tujuan (*goal*) telah tercapai.

Pada *backward chaining*, proses dimulai dari tujuan, yaitu mengecek rule-rule yang memuat tujuan tersebut di bagian aksinya. Jika prekondisi tidak diketahui, maka nilainya diminta ke pengguna atau dicari lewat *rule* lain.

## 2. Frame/Object System

Ide sistem *frame/object* adalah merepresentasikan pengetahuan sebagai kelas-kelas objek yang memiliki *field-field* tertentu. Kaitan antar kelas didefinisikan dengan turunan.

Contoh untuk ide umum objek dapat dilihat pada Tabel II-1.

**Tabel II-1 Contoh property objek untuk "Kucing"**

Object	Property	Values
Cat	is-a	Mammal
	nb-of-legs	4
	has-fur	yes
	habitat	domestic
	size	small
	can-fly	no

Dalam penggunaannya, dilakukan instansiasi objek dengan karakteristik dan kelakuan seperti yang sudah didefinisikan pada kelasnya. Jika dibutuhkan, selain properti yang nilainya sama untuk semua objek sejenis, dimungkinkan property yang nilainya tidak sama. Untuk contoh kucing di atas, dapat ditambahkan property *name* untuk nama kucing tersebut, *property owner*, *color*, dan lain-lain.

Turunan digunakan untuk menghemat penyimpanan data karena turunan mewarisi beberapa sifat dari induknya. Selain itu, turunan memudahkan untuk melihat kaitan antar objek.

Pengembangan sistem berorientasi objek sangat luas. Menurut [PUP93] beberapa diantaranya adalah:

1. Menstrukturkan basis data/basis pengetahuan
2. Penambahan konsep OO pada bahasa pemrograman yang sudah ada.
3. Pengembangan bahasa khusus OO
4. Bahasa-bahasa KLONE untuk klasifikasi objek secara otomatis berdasarkan properti yang sudah didefinisikan
5. *Abstract data type* (ADT) untuk mendeskripsikan fungsionalitas objek tanpa memperhatikan detail implementasi
6. *Semantic net* untuk representasi grafis.

Kelebihan sistem objek adalah mudah untuk merancang dan memahaminya karena sederhana. Selain itu sistem ini dapat merepresentasikan turunan dengan modular. Kekurangannya adalah tidak bisa merepresentasikan negasi, disjungsi, kuantifikasi. Masalah ini bisa ditangan dengan mengembangkan notasi sistem objek, misalnya dengan menggabungkannya dengan sistem rule, dan/atau menambahkan *procedural attachment*.

## 2.4 Ontologi

Salah satu hal yang terkait rekayasa pengetahuan adalah rekayasa ontologi. Definisi ontologi dari sudut pandang AI adalah spesifikasi eksplisit dari suatu konsep yang disetujui bersama [Gruber]. Yang dimaksud konseptualisasi di sini adalah konsep pemodelan ranah pengetahuan secara umum, protokol komunikasi antar agen yang berkerja sama (jika ada), dan representasi dari suatu ranah permasalahan. Ontologi terdiri dari: definisi konsep-konsep, organisasi hirarkis dari konsep (jika ada), hubungan antara konsep-konsep, serta aksioma untuk menformalisasi definisi dan hubungan [GEN87]. Ontologi dirancang oleh perekayasa pengetahuan, dan terdiri dari dua macam, yaitu ontologi khusus dan ontologi umum.

### 1. Ontologi khusus

Ontologi khusus adalah ontologi yang menangani ranah permasalahan yang sempit. Adapun langkah yang dilakukan untuk membuat ontologi khusus adalah sebagai berikut.

- Perekayasa harus mengerti ranah cukup untuk menentukan objek dan fakta mana yang penting dalam permasalahan, dan mana yang dapat diabaikan
- Tentukan predikat, fungsi dan konstanta, yaitu ubahlah konsep-konsep penting pada ranah menjadi formal.
- Representasikan pengetahuan umum dalam ranah
- Representasikan salah satu kasus permasalahan. Jika ontologi sudah dibangun dengan baik, maka langkah ini akan mudah.
- Ajukan *query* dengan menggunakan kasus pada langkah sebelumnya.

### 2. Ontologi umum

Ontologi umum memuat konsep-konsep yang umum, yang bisa diterapkan pada ontologi khusus. Selain itu ontologi umum digunakan untuk menghubungkan berbagai ontologi khusus. Hal-hal yang ditangani oleh ontologi umum antara lain:

- Kategori: Banyak objek memiliki kesamaan. Semua ini dapat dikelompokkan dalam kategori.
- Ukuran: Informasi mengenai massa, umur, harga, dll berhubungan dengan kuantitas yang disebut ukuran.
- Objek gabungan: Objek seperti mobil terdiri dari mesin, ban, dll, yang disusun dengan sedemikian rupa. Ontologi umum menangani bagaimana struktur gabungan direpresentasikan
- Waktu, *space*, dan perubahan: Ontologi umum menangani hubungan antara aksi dan kejadian yang memiliki ukuran waktu yang berbeda dan tempat yang berbeda. Alam semesta mempunyai ukuran waktu dan spasial yang kontinyu.
- Kejadian dan proses: Kejadian bersifat tunggal, sedangkan proses bersifat kontinyu.
- Objek fisik: Objek fisik meliputi benda hidup dan benda mati
- Wujud: Ontologi umum dapat menangani wujud. Misalnya air memiliki tiga jenis wujud. Contoh lain, jus jeruk merupakan wujud lain dari jeruk.
- Objek mental. Objek mental berkaitan dengan agen yang lebih dari satu. Satu buah agen menggunakan pengetahuannya untuk melakukan inferensi. Jika ada agen perlu berhubungan dengan agen lain yang memiliki pengetahuan ataupun ranah pengetahuan lain, maka ia memerlukan model mengenai pengetahuan agen lain, pengetahuan dirinya, kekurangan pengetahuan dirinya, serta prosedur inferensi.