The Role of Ontology in Modern Expert Systems Development

Jason Morris
Morris Technical Solutions LLC
I'm new to AI and my boss wants me to develop a knowledge based classifier software so I have to learn fast.

Based on reading a few papers and websites, I think I will use an expert system and an Ontology. (Am I on the right track?)

I think (but I'm not sure) that I will need both forward and backward chaining.

What tools / languages should I use?

CLIPS, JESS, CLIPS/R2 for the expert systems shell?
OWL for the ontology?

Any suggestions / help is appreciated.
Outline

- Prologue: Setting the Stage
- Part I: Knowledge Engineering
- Part II: Ontology Fundamentals
- Part III: Creating Ontologies
- Part IV: SINFERS Example
- Epilogue: References and Q & A
Prologue: Setting the Stage

Concepts and Context
Key Questions

1. What exactly is a **modern expert system**? What does that entail?
2. Briefly, what is an **ontology**?
3. What is the **relationship** between ontologies and expert systems?

To answer these questions, we require some background. Let’s tackle question #1 now.
## Old vs. New Expert Systems

<table>
<thead>
<tr>
<th>Old</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monolithic</td>
<td>Componentized</td>
</tr>
<tr>
<td>Pre-shell</td>
<td>Shell-derived</td>
</tr>
<tr>
<td>As “oracles”</td>
<td>As services</td>
</tr>
<tr>
<td>Computer-centric</td>
<td>Network-centric</td>
</tr>
<tr>
<td>Non object-oriented architecture</td>
<td>Object-oriented architecture</td>
</tr>
<tr>
<td>“Waterfall” dev</td>
<td>RAD, spiral, and XP dev</td>
</tr>
</tbody>
</table>
Modern Expert Systems

- Built from **OTS/OSS** components.
- Constructed using **shells**.
- Act as an expert **agent** and provide expertise as a **service**.
- Utilize the **internet** as a source of "**common-sense**".
- **Designed** with the latest OOP concepts, RAD/XP practices, and HCI factors.
The AI Value Proposition

Why care about AI/ expert systems at all?

1. Modern businesses need to make complex decisions.
2. Complex decisions require lots of information and applied knowledge.
3. Such decisions must be made quickly and reliably.

By reasoning about information using applied knowledge, expert systems help stakeholders make timely and reliable decisions.
Market Drivers & Enablers

What is driving the apparent renaissance AI?

1. **Hardware Power**
   - Dramatic increases in CPU speed, RAM capacity, storage, etc.

2. **Hardware Cost**
   - Mass-production has lowered technology costs.

3. **INTERNET**
   - An incubator for new technologies and a source for new markets – a “killer-app”!
What is an ontology?

In the AI context, an ontology is:

- A collection of **classes**, their **attributes**, and their **relationships**.
- A **description** or **model** of a domain of **discourse** or knowledge (area of expertise).
- A **vocabulary** for conveying thought and conducting reasoning in a domain.
What is an ontology?

- Ontologies are used to **encapsulate** domain knowledge.
- Artificial Intelligence (AI) applications perform symbolic **reasoning** over those domains.
- Ontologies define the **limits** of symbolic reasoning in AI applications.
Linking ES and Ontologies

Ontologies disambiguate meaning.
Part I: Knowledge Engineering

Moving from Noise to Knowledge
Our Knowledge Engineer

- May or may not be a **domain expert** in own right – but possible.
- Expert **programmer**.
- Software **architect**.
- **Liaison** skills.
- Great communicator.
- Makes raw knowledge **programmable**.
From the Engineer’s POV

Our knowledge engineer would prefer that her project get recognition here...

Rather than here.

2009 SIIA CODiE AWARDS

THE DAILY WTF
Curious Perversions in Information Technology
Knowledge Representation

Typically in ES development, the first design decision is how knowledge will be represented in the system.

Ontologies provide:

- An approximate model or view of the world with respect to a domain.
- The bounds of an intelligent system's knowledge base.
- A foundation for sense-making and reasoning.
Parlez-Vous AI?

From a popular ontology editor help file...

“Instances are the actual data in your knowledge base … If you have to make changes to your class or slot structure after instances have been entered, you may lose some information.”

Anything odd about this?
Noise

Defined as the universe of **all possible invariants**...

...an endless sea of qualitative and quantitative values without a **cognitive pattern**.
Data

Noise is filtered and sampled to separate useful measurements (facts) and to form data.

Thus, in a sense, data is created via our cognitive attention.
Information

Data is analyzed and interpreted, to uncover meaning and relationships, producing actionable information.

Information aids decisions.
Knowledge

- Knowledge is derived from information.
- One application of information is to make decisions.
- When we observe the outcomes of those decisions, we can uncover new data or information, and generate new knowledge.
- Knowledge comes in two main types...
Declarative Knowledge Rules

We can represent them by asserting a fact or facts when certain other facts are present.

These represent the invariants that can be inferred when one or more invariants hold.

We can represent them by asserting a fact or facts when certain other facts are present.
Procedural Knowledge Rules

We can represent them by calling functions when certain facts are present.

**Context**

- **Invariant(s)**
  - =>
  - **Actions(s)**

These represent the actions to take when certain invariants hold.

**IF** fact(s)
**THEN**
call function foo

We can represent them by calling **functions** when certain facts are present.
Declarative vs. Procedural

An example of **declarative** knowledge.

**IF**
- (instance-of ?x THING)
- (composed-of ?x CLAY)
- (composed-of ?x SAND)
- (composed-of ?x SILT)

**THEN**
- (instance-of ?x SOIL)

An example of **procedural** knowledge.

**IF** we have the soil property values \( x_1 \land x_2 \land \ldots x_n \)

**THEN** we can compute the soil property

\[ y_m = f(x_1, x_2, \ldots, x_n) \]
Ontological Commitment

“An **agreement** to use a **vocabulary** (i.e., ask queries and make assertions) in a way that is consistent (but not complete) with respect to the theory specified by an **ontology** ...

An agent **commits** to an ontology if its **observable actions** are consistent with the definitions in the ontology.” – Tom Gruber
Tacit (Implicit) Knowledge

- Most expertise is tacit or implied.
- Tacit knowledge is hard or impossible to quantify or qualify.
- Often the result of extensive experience. Hard to verify.
- Inseparable from original problem context.
The “Bottleneck” Issue

Converting Expert Knowledge To Rules

Elicitation

How can they make each other understand what each knows?
Part I: Ontology Fundamentals

The How, What, Where, When, and Why of Ontologies
Ontological Classes

**CLASS**

Represents a “thing” or concept.

**SLOT**

A data field within a class. Type is optional.

**FACET**

An allowed value for a slot [optional].
### Classes vs. Instances

<table>
<thead>
<tr>
<th>Soil Property</th>
<th>organic_carbon</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>id</strong> INTEGER</td>
<td>id 42</td>
</tr>
<tr>
<td><strong>symbol</strong> STRING</td>
<td>symbol &quot;_6A1&quot;</td>
</tr>
<tr>
<td><strong>value</strong> FLOAT</td>
<td>value 0.06</td>
</tr>
<tr>
<td><strong>stdev</strong> STRING</td>
<td>stdev 0.01</td>
</tr>
<tr>
<td><strong>units</strong> STRING</td>
<td>units &quot;kg/kg&quot;</td>
</tr>
</tbody>
</table>
Specializing vs. Inheriting

**Subclassing** or extending a parent class, superclass, or base-class

**Inheritance** from one or more parent classes
Acrisol and Vertisol are specializations of Soil.

A soil horizon can inherit properties from distinct types.
Ontological Relationships

Relationships connect pairs of classes.
Ontological Relationships

Relationships are modeled as slots.

PEDOTRANSFER_FUNCTION uses-a FUZZY_TRAINING_CLASS

FUZZY_TRAINING_CLASS is-a TRAINING_CLASS

CENTROID has-a
Taxonomy vs. Ontology

**Taxonomies:**
- Usually are a single, hierarchical classification within a subject
- Primarily focused on “is-a” relationships between classes
- Limited in inferencing potential due to lack of relational expressiveness.

**Ontologies:**
- Subsume taxonomies.
- Include attributes with cardinality and restricted values.
- Unlimited relationships between entities.
- Superior inferencing support due to relational expressiveness.
Upper Ontologies

- **Benefit:** Provide common sense.
- **Example:** Cyc Project.
- **Issues:**
  - Web 2.0 (Semantic Web)
  - Is this the key to real AI?
  - Harder to generalize knowledge.
  - Where to stop? Granularity?
  - Overlap and integration?
Implied Ontologies

(deftemplate soil-property
  "A fact describing a soil property"
  (slot symbol)
  (slot value)
  (slot error)
  (slot units))

(deftemplate ptf
  "A fact describing a pedotransfer function"
  (slot symbol)
  (multislot args)
  (slot value)
  (slot units))

Using these templates, what could an expert system reason about implicitly?
How do ontologies help?

Knowledge Layer

- General Inferencing Rules
- Business Rules
- Semantic Search (Web 2.0)

Data Layer

- Entity Relationship Diagrams
- SQL Queries
- Data Exchange

Design Layer

- OOP Object Models
- UML Diagrams

Ontologies
Why create ontologies?

- To share common understanding of the structure of information among people or software agents.
- To enable reuse of domain knowledge.
- To make domain assumptions explicit.
- To separate domain knowledge from the operational knowledge.
- To analyze domain knowledge.

Source: Noy, N.; McGuinness, D. 2001
Why create ontologies?

• To accommodate future domain growth.
• To provide for inter-operability with legacy and future intelligent systems.
• To accommodate complex domains with many classes and relationships.
• To support systems where implicit ontologies are insufficient.
## Problem Solving Methods

Defined as reasoning strategies for solving certain problem types. See Also Generic Tasks.

<table>
<thead>
<tr>
<th>Classical PSM</th>
<th>Key PSM Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Heuristic Classification</td>
<td>William Clancey</td>
</tr>
<tr>
<td>• Generate &amp; Test</td>
<td>B. Chandrasekaran</td>
</tr>
<tr>
<td>• Propose &amp; Revise</td>
<td>John McDermott</td>
</tr>
<tr>
<td>• Cover &amp; Differentiate</td>
<td>Allen Newell</td>
</tr>
<tr>
<td>• Acquire &amp; Present</td>
<td>Robert Wielinga</td>
</tr>
</tbody>
</table>

You design your application ontology to support your problem solving method.
In “Preliminary Steps Towards a Taxonomy of Problem Solving Methods”, McDermott says…

“...In traditional expert systems terminology, a problem-solving method is called an inference engine.”

In our case, we are using a rule-engine as our inference engine. So, forward-chaining through rules is our problem-solving method.
PSMs, Tasks, and Goals

PSM

- goal_1:
  - task_1_1
  - task_1_2

- goal_2:
  - task_2_1
  - task_2_2
  - task_2_m

- goal_n:
  - task_n_1

Rule Engine

- module_1:
  - rule_1_1
  - rule_1_2

- module_2:
  - rule_2_1
  - rule_2_2
  - rule_2_m

- module_n:
  - rule_n_1
Doing this allows developers to make explicit tradeoffs between **reusability** and **granularity** of the required domain knowledge. Separates out **control** and **procedural** knowledge.
Closed World Assumption

If a thing or concept is not explicitly defined in an ontology where CWA holds, then:

- All references to it are logically FALSE.
- You cannot reason about it.

CWA is used when a “best” answer is required despite an incomplete knowledge base.
Linking Rules & Ontology

- Ontological **instances** are the **input** and **output** of expert systems backed by ontologies.
- Rules **operate** on ontological instances.
- The **granularity** of your ontologies determines the **expressiveness** of your **facts** and **rules**.
- The expressiveness of your facts and rules determines the degree to which you can make **inferences** within your problem scope.
Part II: Ontology Creation

How to design and build a domain ontology.
Ontology Truisms

• An ontology is a designed artifact.
• There is no right or wrong way to begin.
• A bottom-up or top-down approach is fine.
• Expressiveness comes from relations not the classes and their slots.
• Separate domain ontologies from control ontologies so that control implementation is free to vary.
Ontology Creation Basics

1. Identify as many “things” and concepts in your domain as you can.
2. For each thing or concept, enumerate its attributes and properties. Specify any restricted values.
3. For each pair of things or concepts, decide if there is some relationship between them. Look for hierarchy, composition, cooperation, and dependence.
Markup Technologies

- RFD - Resource Description Framework
- RDF Schema -
- DAML - DARPA Agent Markup Language
- SHOE - Simple HTML Ontology Extensions
- OIL - Ontology Inference Layer
- DAML+OIL
- OWL - Web Ontology Language
Using an Ontology IDE

- Saves you time and effort **designing**.
- Easy to **export** and **import** your ontology via different formats (i.e., RDF, OWL, etc.).
- Often coupled with an **instance editor** to create a “knowledge-base”.
- Protégé is the defacto OSS standard. [http://protege.stanford.edu/](http://protege.stanford.edu/)
Ontology Languages / Tools

- Ontolingua
  http://www.ksl.stanford.edu/software/ontolingua/
- Loom / Power Loom / Ontosaurus
  http://www.isi.edu/isd/LOOM/LOOM-HOME.html
- SWOOP
  http://code.google.com/p/swoop/
- OntoEdit
  http://www.ontoknowledge.org/tools/ontoedit.shtml
- TopBraid Composer
  http://www.topquadrant.com/topbraid/composer/index.html
Part IV: SINFERS Example

Lessons Learned from the Soil Inferencing System (SINFERS) Project
SINFERS Project

- University of Sydney, AU
- Estimation of soil properties via pedotransfer functions (PTF).
- Use rules to select PTFs.
- Used by soil scientists, farmers, civil engineers, and conservationalists.
SINFERS Design Task

Given an initial set, \(\{P\}\) of \(n\) soil properties, the rule compute-\(\text{ptf-}p^*\) is applied:

IF \(\{q\}: \{q\} \cup \{P\}\) PTF(\{q\})

THEN compute \(p^*\) and add to working memory

\[ p^* = f(\{q\}) \]
SINFERS in Protégé
How Ontology Pays Off

- We avoided hard-coding PTFs as functor-like Java classes.
- We avoided ugly GUI issues via early detection.
- We were able to more rapidly agree on design intent once we had a common vocabulary.
- We can write XSLT's to map our input spec to other Australian government database schema.
Epilogue

References and Q & A
Web Ontology History

- SHOE
- OIL
- DAML
- RDF
- DAML+OIL
- OWL
Famous Ontology Folks

Tom Gruber  Deb McGuinness  Dieter Fensel

Many of their colleagues and students are also great sources of ontology literature.
Take-Away Points

Developing a domain ontology can help:

• Facilitate communication and understanding between knowledge engineers and domain experts.
• Establish relationships and semantics.
• Fulfill Closed-World assumptions.
• Facilitate creation of other design artifacts:
  – UML object models
  – Inference rules
References I

References II

The entire archive of ontology papers used in this presentation is available as a ZIP.

Please ask me or James Owens if you’d like a copy.
Thank you all for your kind attention!!

15 Minutes for Questions