Ontology Building: A Survey of Editing Tools
By Michael Denny

Why Ontologies?

As the hype of past decades fades, the current heir to the artificial intelligence legacy may well be ontologies. Evolving from semantic network notions, modern ontologies are proving quite useful. And they are doing so without relying on the jumble of rule-based techniques common in earlier knowledge representation efforts. These structured depictions or models of known (and accepted) facts are being built today to make a number of applications more capable of handling complex and disparate information. They appear most effective when the semantic distinctions that humans take for granted are crucial to the application's purpose. This may mean handling the common sense lurking in natural language excerpts or the expertise embedded in domain-specific explications and working repositories.

The semantic structuring achieved by ontologies differs from the superficial composition and formatting of information (as data) afforded by relational and XML databases. With databases virtually all of the semantic content has to be captured in the application logic. Ontologies, however, are often able to provide an objective specification of domain information by representing a consensual agreement on the concepts and relations characterizing the way knowledge in that domain is expressed. This specification can be the first step in building semantically-aware information systems to support diverse enterprise, government, and personal activities.

Examples span several areas including: Semantic Web research; the creation of medical guidelines for managing patient health; mapping the genomes of plants and animals; searching for specific public information resources; collaborative engineering design; in-depth security analysis; and the automated exchange of electronic information among commercial trading partners.

Semantic Web

In the Semantic Web vision, unambiguous sense in a dialog among remote applications or agents can be achieved through shared reference to the ontologies available on the network, albeit an always changing combination of upper level and domain ontologies. We just have to assume that each ontology is consensual and congruent with the other shared ontologies (e.g., ontologies routinely include one another). The result is a common domain of discourse that can be interpreted further by rules of inference and application logic. Note that ontologies put no constraints on publishing (possibly
contradictory) information on the Web, only on its (possible) interpretations.

**Kinds of Ontologies**

Ontologies may vary not only in their content, but in their structure and implementation.

**Level of description**

Building an ontology means different things to different practitioners. How one goes about describing something reflects a progression in ontologies from simple lexicons or controlled vocabularies, to categorically organized thesauri, to taxonomies where terms are given distinguishing properties, to full-blown ontologies where these properties can define new concepts and where concepts have named relationships with other concepts, like "changes the effect of" or "buys from".

**Conceptual scope**

Ontologies also differ in respect to the scope and purpose of their content. The most prominent distinction is between the domain ontologies describing specific fields of endeavor, like medicine, and upper level ontologies describing the basic concepts and relationships invoked when information about any domain is expressed in natural language. The synergy among ontologies -- exploitable by a vertical application -- springs from the cross-referencing between upper level ontologies and various domain ontologies.

**Instantiation**

All ontologies have a part that historically has been called the terminological component. This is roughly analogous to what we know as the schema for a relational database or XML document. It defines the terms and structure of the ontology's area of interest. The second part, the assertional component, populates the ontology further with instances or individuals that manifest that terminological definition. This extension can be separated in implementation from the ontology and maintained as a knowledge base. The dividing line, however, between a thing's being treated as a concept and being treated as an individual is usually an ontology-specific decision. Whether the 1965 Ford Mustang GT is an individual Ford automobile, or the vehicle with license plate number AXL429 is an individual Ford (as an instance of the subclass 1965 Ford Mustang GT), may vary between two valid automotive ontologies.

**Specification language**

Ontologies are not all built the same way. A number of possible languages can be used, including general logic programming languages like Prolog. More common, however, are languages that have evolved specifically to support ontology construction. The Open Knowledge Base Connectivity (OKBC) model and languages like KIF (and its emerging successor CL -- Common Logic) are examples that have become the bases of other ontology languages. There are also several languages based on description logic. These include Loom and DAML+OIL, which is currently being evolved into the Web Ontology Language (OWL) standard. When comparing ontology languages, what is given up for computability and simplicity is usually language expressiveness, which isn't always a bad deal. A language need only be as rich and expressive as is necessary to represent the nuance and intricacy of knowledge that the ontology's purpose and its developers demand.

The wide array of information residing on the Web has given ontology use an impetus, and ontology languages increasingly rely on W3C technologies like RDF Schema as a language layer, XML Schema for data typing, and RDF to assert data.

**Building Ontologies**

The basic steps in building an ontology are straightforward. Various methodologies exist to guide the theoretical approach taken, and numerous ontology building tools are available. The problem is that these procedures have not coalesced into popular development styles or protocols, and the tools have not yet matured to the degree one expects in other software practices. Further, full support for the latest ontology languages is lacking.

An ontology is typically built in more-or-less the following manner:

1. **Acquire domain knowledge**
   
   Assemble appropriate information resources and expertise that will define, with consensus and consistency, the terms used formally to describe things in the domain of interest. These definitions must be collected so that they can be expressed in a common language selected for the ontology.

2. **Organize the ontology**
   
   Design the overall conceptual structure of the domain. This will likely involve identifying the domain's principal concrete concepts and their properties, identifying the relationships among the concepts, creating abstract concepts as organizing features, referencing or including supporting ontologies, distinguishing which concepts have instances, and applying other guidelines of your chosen methodology.

3. **Flesh out the ontology**
   
   Add concepts, relations, and individuals to the level of detail necessary to satisfy the purposes of the ontology.

4. **Check your work**
   
   Reconcile syntactic, logical, and semantic inconsistencies among the ontology elements. Consistency checking may also involve automatic classification that defines new concepts based on individual properties and class relationships.

5. **Commit the ontology**
   
   Incumbent on any ontology development effort is a final verification of the ontology by domain experts and the subsequent commitment of the ontology by publishing it within its intended deployment environment.

**Using tools**

Software tools are available to accomplish most aspects of ontology development. While ontology editors are useful during each step outlined above, other types of ontology building tools are also needed along the way.

Development projects often involve solutions using numerous ontologies from external sources as well as existing and newly developed in-house ontologies. Ontologies from any source may progress through a series of versions. In the end, careful management of this collection of heterogeneous ontologies becomes necessary to keep track of them. Tools also help to map and link between them, compare them, reconcile and validate them, merge them, and convert them into other forms. Ontologies may be derived from or transformed into forms such as W3C XML Schemas, database schemas, and UML to achieve integration with associated enterprise applications.
Still other tools can help acquire, organize, and visualize the domain knowledge before and during the building of a formal ontology.

When starting out on an ontology project, the first and reasonable reaction is to find a suitable ontology software editor. It's hoped this broad summary of available editors will give prospective ontology developers a head start.

by Michael Denny

Survey of Ontology Editors

This survey covers software tools that have ontology editing capabilities and are in use today. The tools may be useful for building ontology schemas (terminological component) alone or together with instance data. Ontology browsers without an editing focus and other types of ontology building tools are not included. Otherwise, the objective was to identify as broad a cross-section of editing software as possible. The editing tools are not necessarily production level development tools, and some may offer only limited functionality and user support.

Concise descriptions of each software tool were compiled and then reviewed by the organization currently providing the software for commercial, open, or restricted distribution. The descriptions are factored into a dozen different categories covering important functions and features of the software. These categories appear in Table 1 summarizing the results. (When possibly subtle distinctions in meaning or approach arose in these descriptions, we elected to retain the words of the tool provider.)

Despite the immaturity of the field, we were able to identify a surprising number of ontology editors -- about 50 overall.

Application

Commercial products include standalone editors designed exclusively for building ontologies in any domain, and editors that are part of commercial software suites designed to deliver broad enterprise integration solutions. Other editing software is the outcome of academic and government funded projects investigating the technical application of ontologies. Some editors are intended for building ontologies in a specific domain but still capable of general-purpose ontology building regardless of content focus. These ontology editors may have enhanced support for information standards unique to their target domain. An example in medicine is the OpenKnoMe editor's support of the GALEN reference medical terminology. Editors may also specifically support a broad upper level ontology, as in the case of the editing environment that has grown up around the unique Cyc ontology and is being released under the OpenCyc initiative.

The enterprise-oriented products have mostly started out as data integration tools like those from Unicorn Solutions and Modulant or as content management tools like Applied Semantics' offering. These latter products are more likely to include linguistic classification and stochastic analysis capabilities to aid in information extraction from unstructured content. This information can potentially become instance data or extend the ontology itself.

A few ontology editors included in the survey are actually software specification tools that are sufficiently general purpose to allow construction of domain ontologies. These tools, like Microsoft's Visio for Enterprise Architects, use an object-oriented specification language to model an information domain (in this case, the Object Role Modeling language). These tools presently lack useful export capabilities, although independent tools to convert between UML and ontology languages like DAML+OIL are under development.
Methodology

When ontology technologies emerged in the 1990s, the focus on knowledge acquisition influenced the way new capabilities were put to use in the field. Early ontology editors, for example, adopted the popular KADS method for developing knowledge bases. This orientation is not as evident in today's tools. Indeed, explicit support for a particular knowledge engineering methodology is not common. A few exceptions include Ontology Works' IODE and the Technical University of Madrid's WebODE, both with support for specific ontology organization approaches. There is also increasing support for common upper level ontologies like WordNet, Cyc, and others.

Interoperability

Ontology building today is a fragmented practice. The situation, in part, is a result of the proliferation of logic languages and information models that have combined to yield even more ontology forms and editing environments. These tools and methodologies, along with the ontologies built with them, generally exist without proven interoperability. This is one of the challenges facing the practice along with establishing methods to integrate ontology components with enterprise information systems and standards.

Ontologies are for sharing. They are intended to serve as consensual rallying points to exchange and interpret information. Clearly, the wider the range of applications and other ontologies that can use an ontology, the greater its utility and the mutual utility of the interrelating ontologies. This requires formal compatibility on syntactic levels as well as semantic levels. One consideration in the enterprise realm, for example, is the ability of a domain ontology to accommodate specialized XML languages and controlled vocabularies being adopted as standards in various industries. None of the current ontology editors address this capability.

Interoperability, instead, is being addressed simply through an editor's ability to import and export ontologies in different language serializations. Some tools like Stanford Knowledge Systems Lab's Ontolingua offer a wide range of translations, while most are limited. Importing or exporting ontologies in the newer languages like DAML+OIL usually means that the translation is only partial and expressivity is lost. A few editors like Web ODE also offer heterogeneous ontology merging capabilities.

Usability

In addition to the features already mentioned, ontology editors vary considerably in their overall feel to the user. The present survey did not attempt to compare editors under use, but a few general observations can be put forward. In terms of breadth and variety of features, especially as they relate to interfacing with other information system components, Protégé 2000 from Stanford Medical Informatics offers an editing environment with several third party plug-ins. From a strict ontology language point of view, Ontolingua and OpenCyc offer, or will offer, development environments affording highly expressive and complete ontology specifications. OpenCyc also provides native access to the most complete upper level ontology available (Cyc). Of the editors supporting DAML+OIL, as an important newer language, OilEd appears to offer strong support for composing description logic expressions.

The ability to organize and manage an emerging ontology is key to an editor's usability. Convenient and intuitive presentations and manipulations of an ontology's interlinking concepts and relations is essential. Because many ontology models support multiple inheritance in the concept hierarchies and relation hierarchies, keeping the associations straight is a challenge. The standard approach is the use of multiple tree views with expanding and contracting levels. A graph presentation is less common, although it can be quite useful for actual ontology editing functions that change concepts and relations. The more effective graph views provide local magnification to facilitate browsing.
ontologies of any appreciable size. The hyperbolic viewer included with the Applied Semantics product, for example, magnifies the center of focus on the graph of concepts (without labeled relations). Other approaches like the Jambalaya plug-in for Protégé-2000 achieve a kind of graphical zooming that nests child concepts inside their parents and allow the user to follow relations by jumping to related concepts. Some practitioners however, such as GALEN users, indicate a preference for non-graphic views for complex ontologies.

Finally, it is worth considering the inferencing support afforded by the ontology editor (beyond classification in description logic editors). While ontologies themselves can be treated as standalone specifications, they are ultimately used to help answer queries about a body of information. Some editors incorporate the ability to add additional axioms and rules of inference to the ontology for evaluation within the defined target of the development environment. For now, rule extensions are mostly proprietary in that standard rule languages able to reference ontology terms and structures directly are not available. A likely candidate to be supported in future ontology editors is RuleML.

**Resources**


