A Collaborative Development Environment for Ontologies (CODE)

Pat Hayes, Raul Saavedra
Institute for Human & Machine Cognition
40 South Alcaniz Street
Pensacola, FL 32501
{phayes,rsaavedra}@ihmc.us

Thomas Reichherzer
Computer Science Department
Indiana University
150 S. Woodlawn Ave
Bloomington, IN 47405
treichhe@cs.indiana.edu

Submitted to Semantic Integration Workshop, October 2003, Sanibel Island.

ABSTRACT

Resolving conceptual conflicts between formalized ontologies is likely to become a major engineering problem as ontologies move into widespread use on the semantic web. We believe that in the immediate and medium-term future, conflict resolution will require the use of human collaboration, and cannot be achieved by automated methods except in simple cases. We are developing an integrated suite of software tools to provide for concept search, collaborative ontology composition and editing, and re-use of existing Web ontologies, based on the *CmapTools* collaborative Concept Map software in widespread use in education, training and knowledge capture applications. Concept maps provide a natural way to display and examine the structure of an ontology in a collaborative setting.

Keywords

Concept Maps, Collaboration, DAML, Ontology, OWL, RDF.

INTRODUCTION

Concept maps [3] or Cmaps, are a graphical representation for simple facts in the form of node-arc-node diagrams. In spite of, or perhaps because of, their simplicity, they have been used successfully in many educational settings as a technique for teaching conceptual thinking, as a knowledge-acquisition methodology and as an input modality for knowledge-acquiring software. The ease with which users of many ages, educational backgrounds and levels of technical expertise learn and use this style of representation is noteworthy; as is the fact that node-arc-node diagrams provide a natural way to display an RDF triple store, and RDF is the 'base' language of the Semantic Web architecture [3]. Noting this convergence suggested to us the possibility of using Cmap software to display and edit Web ontologies.

VIEWING ONTOLOGIES AS CONCEPT MAPS

We anticipate that the semantic web of tomorrow will provide a large set of general-purpose, domain-specific and standardized ontologies that will enable users to rapidly build their own ontologies by taking advantage of existing and agreed upon definitions of concepts and their properties. In this vision, there will be many more 'knowledge engineers' than there are at present, but this skill - which is presently considered somewhat arcane and specialized - will become simpler and more routine, rather as producing an HTML web page is now within the competence on millions of users worldwide. Nevertheless, there is clearly a need to display, search and manipulate ontologies in a form less forbidding to read than, say, RDF/XML.

The evolving W3C standards for ontology description on the Semantic Web comprises a suite of languages of increasing expressive power and complexity: RDF, RDFS, N3, DAML+OIL and OWL. All of these languages can be represented in RDF graph syntax, which consists of sets of triples of the form <subject, predicate, object>; the stronger semantic conditions of the subsequent languages can be viewed as *semantic extensions* imposed on particular RDF vocabularies (typically indicated by a URI Qname prefix, as for example in *rdfs:subClassOf* or *owl:sameAs*.) It is natural to display this syntax in a node-arc-node graphical format, which is indeed used in the specification documents themselves; and it is straightforward to apply graph-layout algorithms to generate a graphical rendering, in the form of a concept map, of any Web ontology represented as an RDF semantic extensions.

We found however that a straightforward graphical rendering of real RDF ontologies is often unreadable in practice, for a number of reasons. First, RDFS and OWL ontologies often supply full type information, which generates a lot of graphical clutter, with a few common 'type' nodes being densely linked to many other nodes in the graph; such information is useful for machine processing but redundant and distracting for human readers. Second, several of the more complex languages use constructions such as lists which are rendered into triples; these 'structural' triples tend to be distributed among the more contentful triples and fail to convey the intended meaning in a visually convincing fashion; and finally, the ontologies often contain typed literals, comment strings and other structures which are hard to follow when rendered graphically from the RDF syntax. For all these reasons, we have designed a special-purpose concept map tool which lays out Web ontologies more 'naturally', so as to reveal the essential content of the RDF graph while displaying lists as single nodes, literals and text in natural ways, and with all the 'obvious' typing information hidden from view in the graphical display. Together with several other graphical techniques, such as context-sensitive zooming, the resulting software is capable of automatically generating readable graphical layouts for DAML+OIL and OWL ontologies represented as RDF graphs containing many hundreds of concepts.

One of our main goals is to let the knowledge engineer see and edit the ontology as what the ontology really is in underlying structural terms: a graph. This ontology graph is created from the set of RDF triples defining the ontology. Now there we encounter the issue of where to place those ontology elements in space for visualization purposes. Ontology syntax and language(s) do not provide any placement information associated to any of its elements, so an automatic layout algorithm is therefore needed to place those graph elements over a canvas, and then connect those elements on the canvas to mirror the way they are connected in the ontology. The algorithm chosen was a variant of Sugiyama's algorithm for laying out hierarchical graphs [4]; this seems appropriate since many ontologies have a hierarchical subclass/type structure.

Some triples don't have to be directly represented in the Cmap. For example, comments can become just attributes of the node in the graph associated to the subject the comment applies to. In general, all objects that never appear as subjects in any triple can be hidden this way, making them attributes of the subject, without any effect on the underlying structure of the graph. "Type" triples are processed in a similar fashion. For instance, a triple of the form [A, rdf:type, T], can be also "hidden", keeping the type T information still accessible by making it just an attribute of the node A in the final graph. This reduces the graph density dramatically. If type triples are fully represented in the final graph, many of the type nodes T would become sinks of very many incoming edges, thus unnecessarily cluttering the graph, and making it harder to lay it out and visualize. We provide this hiding feature as a customizable option, so the user can choose what predicates or objects in the triples are to become attributes of the subject node, instead of separate nodes in the final graph.

Another important simplification is the replacement of all URI's and URL's with Qnames. For example, the URI http://www.w3.org/1999/02/22-rdf-syntax-ns#type gets transformed into "rdf:type". This basically reduces the length of all nodes on the final graph, making them easier to read, and also the graph becomes easier to navigate and to visualize as a whole.

After those attribute level simplifications are done on the graph, another stage in the graph transformation process identifies higher level constructs that involve several RDF triples. For example, instead of showing all the actual RDF nodes that constitute a daml list, the code collapses all those into just one special node labeled with the list of elements, and that node is connected to each of the separate element nodes contained in the list. DAML restrictions are identified and processed in a similar way.

The screen shots in Figure 1 illustrate the improvements in readability achieved by these methods. In a large Cmap the effects are even more marked. The software can also be used to edit or compose RDF ontologies by simple click-and-drag operations on the graphical display, and by select-and-paste operations on subgraphs of existing ontologies. This style of composition and editing of concept maps has proven useable by people with little technical background or special training. The 'natural' style of ontology display preserves the intuitive properties which make Cmaps useful.

Although Cmaps are used as the primary viewing and editing format for ontologies, CODE users can also view ontologies as lists of triples ordered in various ways, or as RDF/XML code; the software tracks these various views and maintains internal coherence, so that for example if a concept is found in the triples viewer – which often provides for a more rapid scan of the concepts in an ontology - then the Cmap window will be automatically centered to that place in the graph. Several studies have shown that the spatial metaphor of *location* of a node in a graph and of *navigating* through a graph is critical to the success of the Cmap user interface, particularly for large-scale graphs constructed and edited over an extended period (i.e. more than a single work session).

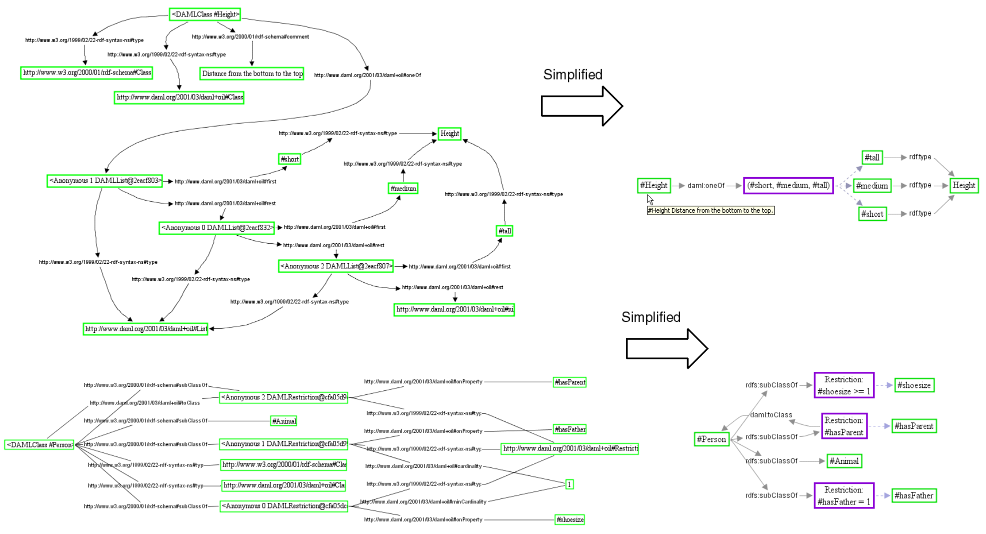


Figure 1: Original RDF graphs for Lists and DAML restrictions, and the simplified versions (right)

CMAPTOOLS AND COLLABORATIVE WORK

Our software is based on *CmapTools* [1](see also http://cmap.coginst.uwf.edu/), an integrated system for collaborative editing, storing and manipulating Cmaps which has been downloaded by users in approximately 150 countries. A similar Cmap interface based on that used by CmapTools was incorporated into the SHAKEN knowledge-entry system [2]; an application closely related to the one described here. Other applications are listed in the appendix below.

CmapTools supports collaboration among individuals at three different levels. First, users can collaborate by sharing sets of concept maps on public servers with controlled access. Second, users from different locations can start a session to synchronously edit a concept map, viewing simultaneously changes in the map as they are performed by the participants; the software is based on a comprehensive real-time agent environment which can maintain synchronicity globally, using internet protocols. And third, users can asynchronously attach notes and threads of discussion to map constituents. All of these functions adapt smoothly to the ontology tool suite we have implemented.

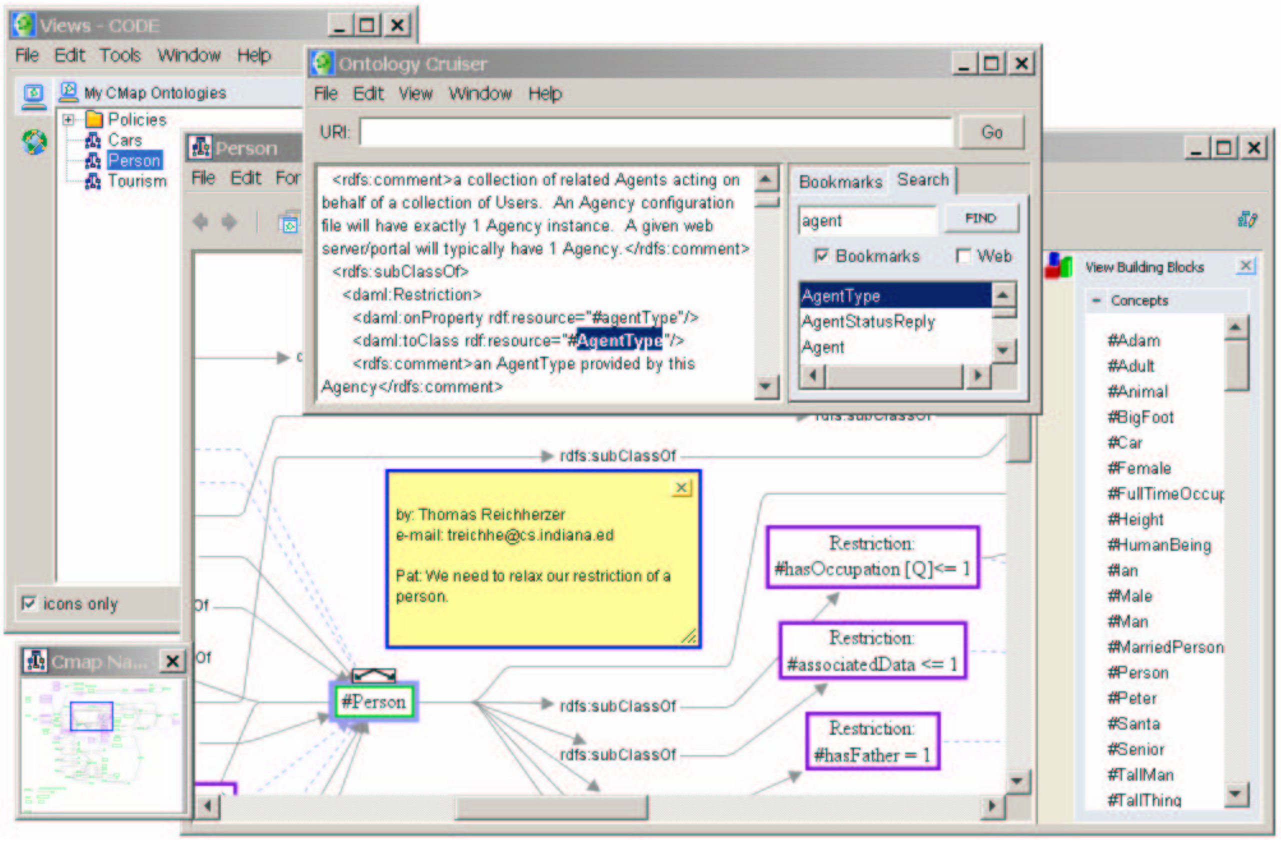
Among the servers in the *CmapTools* architecture, Cmap Servers are primarily responsible for making knowledge models publicly available across geographically-distant sites. They also enable users to collaborate on a knowledge model both synchronously and asynchronously. In support of asynchronous collaboration, servers facilitate discussion threads in concept maps and access control on the knowledge models. For synchronous collaboration, servers support simultaneous editing of concept maps by multiple users from different sites.

Cmap servers register themselves with a designated directory server in the *CmapTools* network that keeps track of available servers and services. Clients use the register of the directory server to present a list of places to the users where ‘knowledge models’ in the form of annotated Cmaps are published. While users can join the existing public *CmapTools* network with their own servers, they can also set up a private network that protects Cmap servers from becoming visible to others. Another type of server in the *CmapTools* architecture is the index server. Its purpose is to facilitate search capabilites enabling users to find resources.

Ontology Cruiser

In addition to the ontology-adapted Cmap editing and composing interface, CODE contains facilities for searching for ontologies and concepts described by existing ontologies.

To support knowledge engineers in building their own ontologies, we have developed a tool called the Ontology Cruiser that is integrated into the Cmap ontology editor. The tool provides a graphical user interface for browsing and bookmarking locally stored ontologies and ontologies from the web as well as searching for concepts within bookmarked ontologies or ontologies indexed by publicly accessible search engines. The tool’s interface is designed similarly to an HTML browser, allowing users to view ontologies from the web or to manage frequently used and locally stored ontologies by means of bookmarking. However, unlike an HTML browser that enables users to view and navigate HTML pages, our focus has been to design the Cruiser such that it is particularly helpful in finding concepts within ontologies that may serve as the building blocks of new ontologies. To achieve our design goal, we focused on two aspects: (1) First, we allow knowledge engineers to browse ontologies in addition to XML text format as a set of RDF triples, or in the form of a concept map automatically generated from the original XML text. For the additional formats, we provide similar navigation and search aids as for the editor to assist knowledge engineers in dealing with large ontologies and finding concepts of interest. (2) Second, we index concepts in bookmarked ontologies and provide a query interface to our locally stored index as well as public search engines for DAML and OWL. The Cruiser displays concepts that match a search query and downloads and depicts the corresponding ontology of a concept when selected in the interface. The concept index for bookmarked ontologies is kept in synch with the bookmarks to assure that concepts in user selected ontologies will be available for building new ontologies. Figure 2 shows a screenshot of the Ontology Cruiser, with two ontologies in view: the DAML'guide' ontology http://www.daml.org/2001/03/daml+oil-ex.daml as an autogenerated Cmap, and the Cyc upper ontology viewed in XML.



```
<rdfs:comment>a collection of related Agents acting on
behalf of a collection of Users.  An Agency configuration
file will have exactly 1 Agency instance.  A given web
server/portal will typically have 1 Agency.</rdfs:comment>
  <rdfs:subClassOf>
    <daml:Restriction>
      <daml:onProperty rdf:resource="#agentType"/>
      <daml:toClass rdf:resource="#AgentType"/>
      <rdfs:comment>an AgentType provided by this
Agency</rdfs:comment>
```

by: Thomas Reichherzer
e-mail: treichhe@cs.indiana.ed

Pat: We need to relax our restriction of a person.

Figure 2: A Screenshot of the Collaborative Ontology Development Environment (CODE).

Building Ontologies

In our methodology for building ontologies, we envision knowledge engineers to start building new ontologies by searching through existing ontologies to collect a set of initial concepts that serve as the building blocks for the new ontology. These building blocks can be collected using the Ontology Cruiser. The Cruiser supports a simple drag-and-drop interface to drag concepts that were identified in a search into a concept map's side panel that is specifically designed for collecting concepts to be used for a new ontology. The side panel indicates graphically which concepts have been included into the new ontology. Knowledge engineers can navigate from the included concepts to the position where they occur in the concept map window by selecting the corresponding concept in the side panel. To insert a concept from the side panel, CODE supports a simple drag-and-drop interface that creates automatically a node labeled by the concept that was dragged into the map area.

CODE in Support of Collaborative Ontology Construction

Although the number of users who are actively engaged in writing Web ontologies has not yet grown to the point where one can make firm observations, we can predict several possible scenarios where concept integration issues arise and where the collaborative nature of the CODE framework could be useful.

In large-scale team efforts, the Cmap interface is of proven utility in maintaining a conceptual 'picture' of the evolving ontology over an extended development period, and the annotation and communication techniques supported by CODE (illustrated in Figure 2) can be used to focus group effort on concepts.

An 'ontology help service' could offer users attempting to write simple semantic markup detailed help by linking to the user's CODE tool, navigating and editing their ontology in real-time collaborative mode. This would enable the user to 'look over the shoulder' of the help service while the operations are being performed. Other Cmap applications have found this to be a powerful teaching and training device, particularly in concert with another communication channel (typically, a telephone contact).

The *CmapTool* software underlying CODE records every screen event in order to maintain real-time collaboration. The same technology can be used to 'record' a complete history of the ontology editing and construction process which can be replayed and analyzed off-line. We anticipate that this can provide a useful way to discover and correct conceptual errors during ontology construction.

The indexing and archiving provided by the ontology cruiser, joined to the Cmap viewing and navigation abilities, allow users to rapidly survey concepts in available ontologies and see them in context, as the example of figure 2 and to access other resources which have been attached to concepts, including comments, discussion threads, and links to other resources (such as movies, images, web pages, etc.) In this way, the Cmap tools substrate allows ontologies to be presented fully linked to a wide range of clarifying text and informative documentary supporting material. Experiments in using conventional Web search processes to discover useful concepts for informal Cmaps [6] suggest that this technique integrates well with existing Web search technology

APPENDIX

The *CmapTools* and the accompanying knowledge elicitation methodology have been used successfully for capturing, representing and sharing expertise in a variety of domains. In addition to many thousands of educational users, applications include a nuclear cardiology expert system; a prototype system to provide performance support and just-in-time training to fleet Naval electronics technicians [5]; a knowledge preservation model on launch vehicle systems integration at NASA [7], a large-scale knowledge modeling effort to demonstrate the feasibility of eliciting and representing local meteorological knowledge undertaken at the Naval Training Meteorology and Oceanographic Facility at Pensacola Naval Air Station, (http://www.coginst.uwf.edu/projects/STORMLK/) and a large multimedia knowledge model on Mars (http://www.cmex.arc.nasa.gov), constructed entirely by a NASA scientist, without the participation of knowledge engineers. Concept maps have provided a successful interface for subject matter experts to input knowledge to a computer system which automatically generates formal representations without the aid of knowledge engineers [2], see also http://www.ai.sri.com/project/SHAKEN.

ACKNOWLEDGMENTS

This research was supported in part by DARPA under contracts 2507-225-22 and 2507-221-22. We also thank the CmapTools development team for their many contributions to this project.

REFERENCES

1. Cañas, A. J., Ford, K. M., Novak, J. D., Hayes, P., Reichherzer, T., Suri, N., Online Concept Maps. The Science Teacher, 68, 4 (April 2001), 49-51.
2. Clark, P., Thompson, J., Barker, K., Porter, B., Chaudhri, V., Rodriguez, A. Thomere, J., Mishra, S., Gil, Y., Hayes, P., Reichherzer, T. (2001). Knowledge Entry as the Graphical Assembly of Components, Proceedings of the First International Conference on Knowledge Capture (K-Cap'01), pp. 22-29.
3. Novak, J. D., Gowin, D. B., Learning How to Learn. Cambridge University Press, Cambridge, UK (1984).
4. K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical systems. IEEE Trans. Syst. Man Cybern., SMC-11(2):109-125, 1981.
5. Cañas, A. J., J. W. Coffey, T. Reichherzer, N. Suri, R. Carff and G. Hill, El-Tech: A Performance Support System with Embedded Training for Electronics Technicians, Proceedings of the Eleventh Florida Artificial Intelligence Research Symposium, Sanibel Island, Florida, May 1997.
6. Cañas,A. J., M. Carvalho, M. Arguedas, Mining the Web to Suggest Concepts during Concept Mapping: Preliminary Results , XIII Simpósio Brasileiro de Informática na Educação – SBIE – UNISINOS 2002, November 2002, Brazil.
7. Coffey, J. W., R. Hoffman, A. J. Cañas and K. M. Ford, A Concept Map-Based Knowledge Modeling Approach to Expert Knowledge Sharing, IKS 2002- The IASTED International Conference on Information and Knowledge Sharing, November 2002, Virgin Islands.
8. Resource Description Framework (RDF), W3C Recommendation 22-February-1999; http://www.w3.org/TR/REC-rdf-syntax/.