Dynamic Ontological Support for Qualitative Reasoning in The Knowledge Collective (TKC)

Jay Yusko and Martha Evens
Illinois Institute of Technology
Department of Computer Science
10 West 31st Street, Chicago, IL 60616
jay.yusko@gensolco.com, evens@iit.edu

Abstract

If you want to do Qualitative Reasoning about a specific domain, ontologies along with an ontology inference engine are needed to support that reasoning. If you want to reuse information between multiple systems, you have to share knowledge and ontologies that are needed for each system. If you also want to have low maintenance of the ontologies, the ontologies have to be developed dynamically as needed by the end user during a session with The Knowledge Collective (TKC). If agents bring along their own ontologies instead of having an ontology server or ontology agents, the ontology for a task-specific agent is easier to maintain and develop. TKC framework uses objects to represent the ontologies and a production rule ontology inference engine for Qualitative Reasoning about domain specific ontologies.

Introduction

A group of students and faculty at Rush Medical College and Illinois Institute of Technology have been building medical learning systems for the last twenty years. Even though much of the knowledge underlying these systems is the same, they have laboriously constructed a new knowledge base for each new system. The goal of The Knowledge Collective (TKC) is to provide a reusable, extensible knowledge base that can be used to support a wide variety of systems.

TKC is an intelligent knowledge base that is built on a multi-layered, multi-agent framework with the goal of supporting Qualitative Reasoning (Forbus 1996) about specific domains. This framework is described in The Knowledge Collective Ontology section. The reason for this type of framework is so that the knowledge needed by an end user to do Qualitative Reasoning can be stored in a format that makes sense for the knowledge, and the end user does not need to know where the knowledge is stored or how to access it. Storing and accessing knowledge is the mission of the agents. Therefore an agent needs two kinds of information to understand how the knowledge that the end user wants is stored and how to retrieve it: the meta knowledge (Yusko and Evens 2002) and the ontology of the knowledge base. The meta knowledge is the knowledge that the agent needs to understand its structure and storage of the information. The ontology tells how the agent can make use of the knowledge for the end user. This paper will just cover the ontology side of the equation. If you want to know more about the meta knowledge side, see Yusko and Evens (2002).

As with any agent based system, ontology is very important. Agents have to have an understanding of the environment that they are working in. If an agent is to be the keeper of specific knowledge then the agent needs to know how the semantics of the knowledge is structured. The ontological model can be either artifact based or process based. This is the model that is given to the computer to understand the knowledge (Yusko 1994, Bredeweg and Forbus 2003, Falkenhainer and Forbus 1988). With this type of ontological framework, Qualitative Reasoning can be accomplished for a specific domain.

Ontology architectures will be discussed in the Ontology Background Section. TKC will be described in The Knowledge Collective Ontology Section. A special type of agent called a MicroDroid is used in TKC and will be discussed in the MicroDroid Ontology in The Knowledge Collective Section. Then a medical tutoring system will be used as an example in the Ontology Example section. The paper is summarized in the Conclusion Section.

Ontology Background

Don Hutcheson (2003, p. 45) defines ontology as "a list with relationships to other lists". The foundation for Intelligent Physical Agents (FIPA) describes ontology in the following way (FIPA 2002a p. 34):

> An ontology provides a vocabulary for representing and communicating knowledge about some topic and a set of relationships and properties that hold for the entities denoted by that vocabulary.

An ontology is a model of a specific domain that can be used for Qualitative Reasoning about either structural objects and their relationships or processes using the Qualitative Process Theory of Kenneth Forbus (1985). The ontologies enable agents to communicate with each other and an end user in a intelligent manner. FIPA has a specification for an ontology service (FIPA 2002b). This specification assumes that the system has an ontology server. It talks about using ontology agents to make the ontology available to all agents in the system (FIPA 2002b p. 1-8).

The concept of an ontology server with ontology agents does not properly fit The Knowledge Collective framework. Each time a user connects to TKC, a session is set up. This session will last until the end user has completed the desired tasks. The important issue dealing with the ontology for TKC is that each user session will dynamically build its own ontology. Each user session could have a different ontology depending on what the user is doing.

Many ontologies are stored physically in frames (FIPA 2002b p.16). The thought was originally to store TKC ontologies in frames. Reasoning about frame based ontologies in agent systems is usually done using predicate logic such as prolog type rules. TKC will use production type rules as an ontology inference engine to accomplish Qualitative Reasoning about specific domains as explained the MicroDroid Ontology in The Knowledge Collective Section. How the ontological support works in TKC is explained in the next two sections.

The Knowledge Collective (TKC) Ontology

TKC is a multi-layer multi-agent framework for the reuse of information in an intelligent knowledge base (Yusko In Progress). It contains 6 layers:

1. Graphical User Interface
2. Coordination
3. Application
4. Solution
5. Task
6. Database

Each layer is composed of classes of MicroDroids, a specific type of agent that is explained in the next major section. Therefore, there can be many instances of each MicroDroid in Figure 1. The actual MicroDroids will be discussed in the next section

1. Graphical User Interface Layer

The Graphical User Interface Layer is the portal into the actual application information in TKC. This is an application interface for the user. It gives the end user access to application information. It gives the developer the ability to add, delete, maintain or monitor the MicroDroids in each layer. It also allows the subject matter experts to view, add and update their subject areas.

There is a USER INTERFACE MicroDroid in the Coordination Layer that controls the Graphical User Interface. It can control many screens and all the information about them.

2. Coordinator Layer

The Coordinator Layer controls TKC. This layer always contains three specific purpose MicroDroids: COORDINATOR, USER PROFILE and USER INTERFACE. The COORDINATOR works with the User Interface to deal with all tasks from the Graphical User Interface Layer whether they come from the end user or a developer. The COORDINATOR also sets up a common goal that starts a session. The rest of the MicroDroids in TKC cooperate to satisfy this goal. The COORDINATOR knows about all of MicroDroids in TKC by asking them for information. If MicroDroids are added or deleted, or their functionality is changed, the COORDINATOR will know about these changes. It sends out orders along with the goals

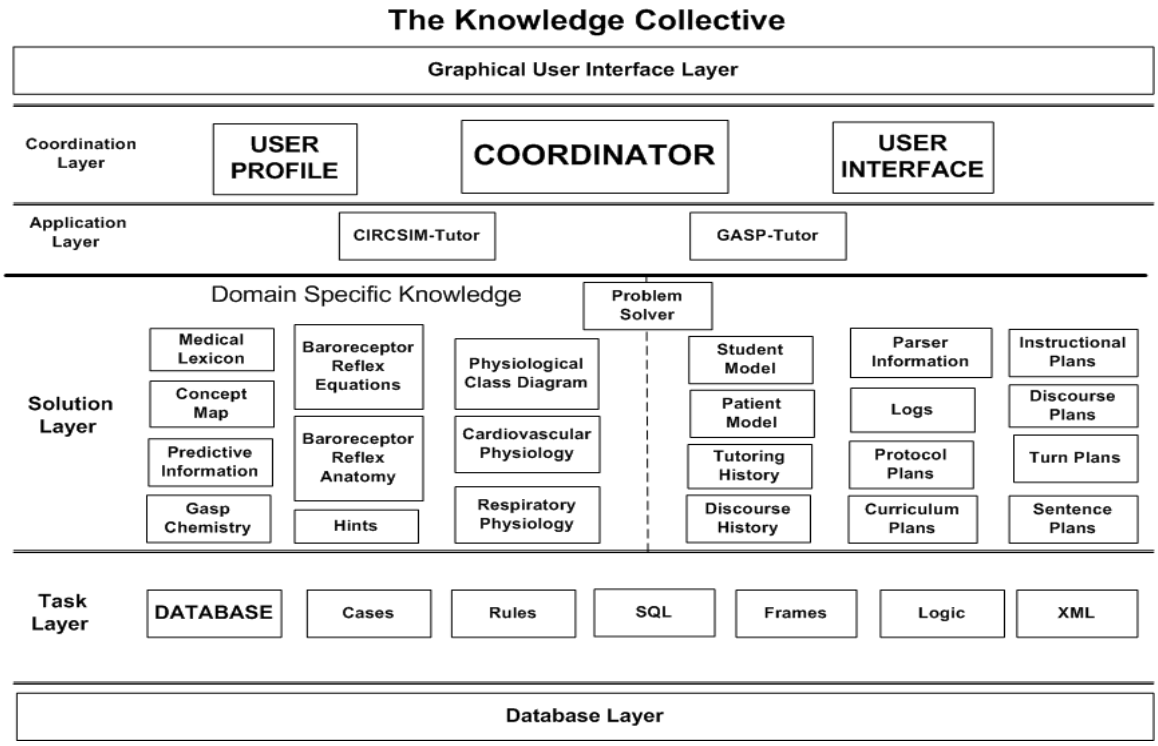

Figure 1: The Knowledge Collective Framework

to find one or more MicroDroids in the Application Layer to solve end user problems. However, the COORDINATOR can interface with any MicroDroid in TKC to solve various development and maintenance problems.

The USER PROFILE can be either an end user or a developer. It controls the information about a specific user.

3. Application Layer

The Application Layer contains a MicroDroid for each application in TKC. These MicroDroids get their orders from the COORDINATOR. They work individually to solve application problems. They use from one to many MicroDroids in the Solution Layer to access, delete or insert application data in the Database Layer. This is the only layer that is application specific. The Knowledge Collective example in Figure 1 is designed to support two medical tutoring systems: CIRCSIM-Tutor and GASP-Tutor.

CIRCSIM-Tutor (Michael et al 2003) deals with the baroreceptor reflex, the part of the circulatory system that controls the blood pressure. GASP-Tutor is a new tutoring system (still being implemented) that deals with the pulmonary system with a focus on the two interacting negative feedback systems that control breathing and gas exchange in the lungs.

4. Solution Layer

There are many MicroDroids in the Solution Layer. Each one knows how to deal with a specific domain as seen in Figure 1. They get their marching orders from an Application MicroDroid. They can work individually or as teams to solve an application problem. They use from one to many MicroDroids in the Task Layer to access, delete or insert application data in the Database Layer.

These MicroDroids are very information specific. If you add a new Application MicroDroid that will use the same solution information, the same Solution MicroDroid will

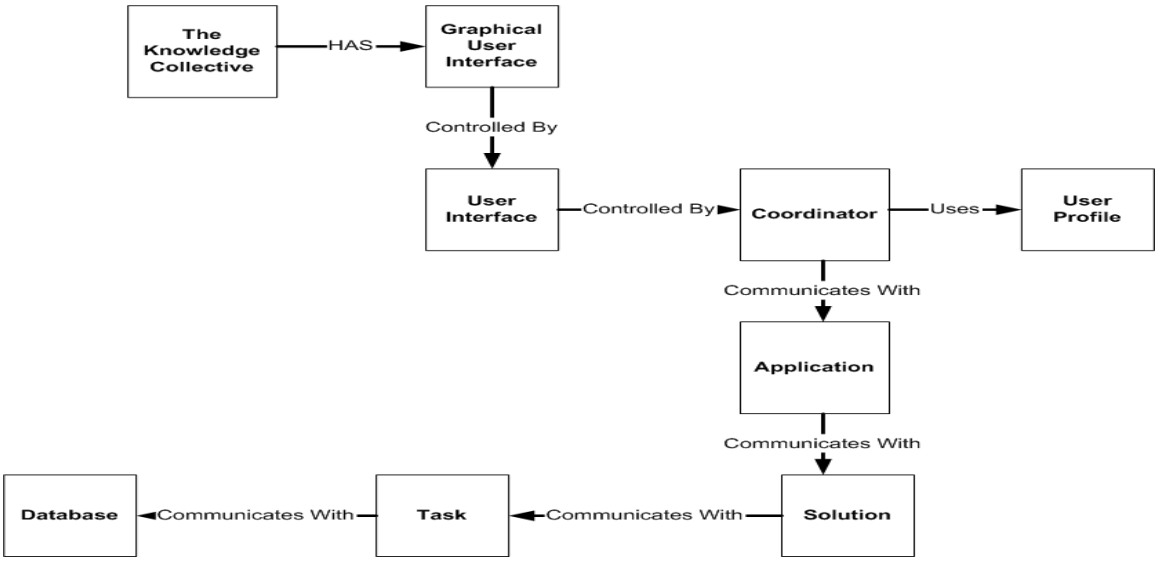

Figure 2: The Knowledge Collective Ontology

respond. If changes are made to the application data structure these MicroDroids will not have to be modified. There are specific MicroDroids that can deal with domain specific information and others that deal with application specific information. For instance CIRCSIM-Tutor uses all the MicroDroids in this layer except Gasp Chemistry and Respiratory Physiology. GASP-Tutor will use all the MicroDroids in the layer except Baroreceptor Reflex Equations and Baroreceptor Reflex Anatomy. This can be seen in Figure 1. This really shows the knowledge and ontology reuse capabilities of TKC.

5. Task Layer

There are many MicroDroids in the Task Layer. Each one knows how to solve specific tasks. They get their marching orders from the Solution Layer. They work individually or as teams to deal with application data in the Database Layer to answer the calls from the Solution Layer.

The MicroDroids in this layer are not application specific. They are data architecture specific. The only time you would add a new one or modify an existing one is if there is a change to the actual data structure. An example would be the Frame MicroDroid. It uses the Frame Building Language (Yusko 1984), which understands how to deal with frames and relationships like semantic networks. Even though the application data is stored in relational format in a relational database in the Database Layer, this MicroDroid knows how to interpret it properly.

However, there is one specific MicroDroid, DATABASE that knows all the information about the Database Layer.

6. Database Layer

The Database Layer can contain from one to many databases. However, there is a DATABASE MicroDroid in the Task Layer that knows about all the databases in this layer and how to access them. It knows about all the databases and what they contain. Therefore there can be connectivity to many databases and only one place to maintain the information about the databases. This layer is where the information used by the MicroDroids is stored. The database is an industry standard relational database.

These six layers make up the overall high level ontology for TKC as shown in Figure 2. This is an object ontology used for doing Qualitative Reasoning about The Knowledge Collective in general.

MicroDroid Ontology in The Knowledge Collective

If you look at Figure 3, MicroDroids are a subclass of Task-Specific Agents based on Franklin and Graesser's Agent Taxonomy (1996, pp. 23). Each box inside of TKC in Figure 1 is a class of MicroDroids that do a very specific task.

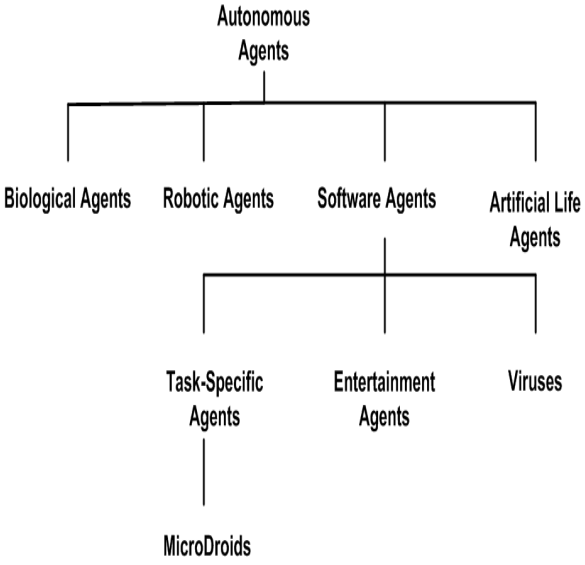

Figure 3: Agent Hierarchy

What really makes a MicroDroid different from the Task-Specific Agents is the fact that MicroDroids do not depend on an ontology server or an ontology agent. Built into every MicroDroid are its own ontology and an ontology inference engine. This is vital since the ontology used during an end user session needs to dynamically build the session ontology. This also allows each MicroDroid to do Qualitative Reasoning about its specific domain using an ontology inference engine.

Most systems physically use frames to develop the ontologies and predicate logic to do the Qualitative Reasoning. It was decided for TKC that objects developed using the Unified Modeling Language (UML) from the Object Management Group (OMG) would be used instead of frames for storing the ontologies. This follows OMG's Model Driven Architecture™ (Frankel 2003) approach. The UML objects are converted to Java classes. The ontology inference engine being used is a Java class that reasons about Java objects and their relationships. The development environment is Eclipse (Gallardo et al., 2003) and the Eclipse Modeling Framework plug-in (Budinsky et al., 2004)). ILOG JRules™ is used to implement the production rule based ontology inference engine. Physically using objects instead of frames, does not preclude the use of concepts like frame semantics as used in FrameNet (Fillmore and Baker 2001). The ontology inference engine will be able to make use of the information.

Ontology Example

When a user logs into TKC, the user is working thru the USER INTERFACE MicroDroid. The USER INTERFACE MicroDroid is always connected to a COORDINATOR MicroDroid and to a USER PROFILE MicroDroid. These three MicroDroids are a part of all sessions and supply the initial ontology for the session. Then the session ontology grows as new MicroDroids are added to the session.

The USER PROFILE MicroDroid sets up the user ontology so that the system can understand the user. The system then has to set up the subject area ontology. This is the purpose of the COORDINATOR MicroDroid to ascertain what the user wants to do. If the user wants to understand circulatory chemistry issues, then the COORDINATOR MicroDroid broadcasts the request to the Application Layer. The GASP-TUTOR MicroDroid would answer and a session pipeline would be set up with the COORDINATOR MicroDroid for the user. The COORDINATOR MicroDroid communicates with the user via the USER INTERFACE MicroDroid. If the user wants to learn about baroreceptor reflexes, the CIRCSIM-Tutor MicroDroid would answer and a pipeline would be set up. If no MicroDroid answers, a list of possible applications would be given to the user.

If the CIRCSIM-Tutor MicroDroid answers, a session ontology is set up. In this case the CIRCSIM-Tutor MicroDroid, as shown in Figure 4 (Evens and Michael In Progress), has a process ontology. Every time a new MicroDroid answers, a new piece of the session ontology is produced. Each MicroDroid understands the environment it works in and adds a piece of the ontology to the session ontology. Therefore, the ontology for the session is developed dynamically as a new MicroDroid becomes part of it. When a session is complete, the MicroDroids involved drop out and take their ontologies with them. Therefore, the ontology of

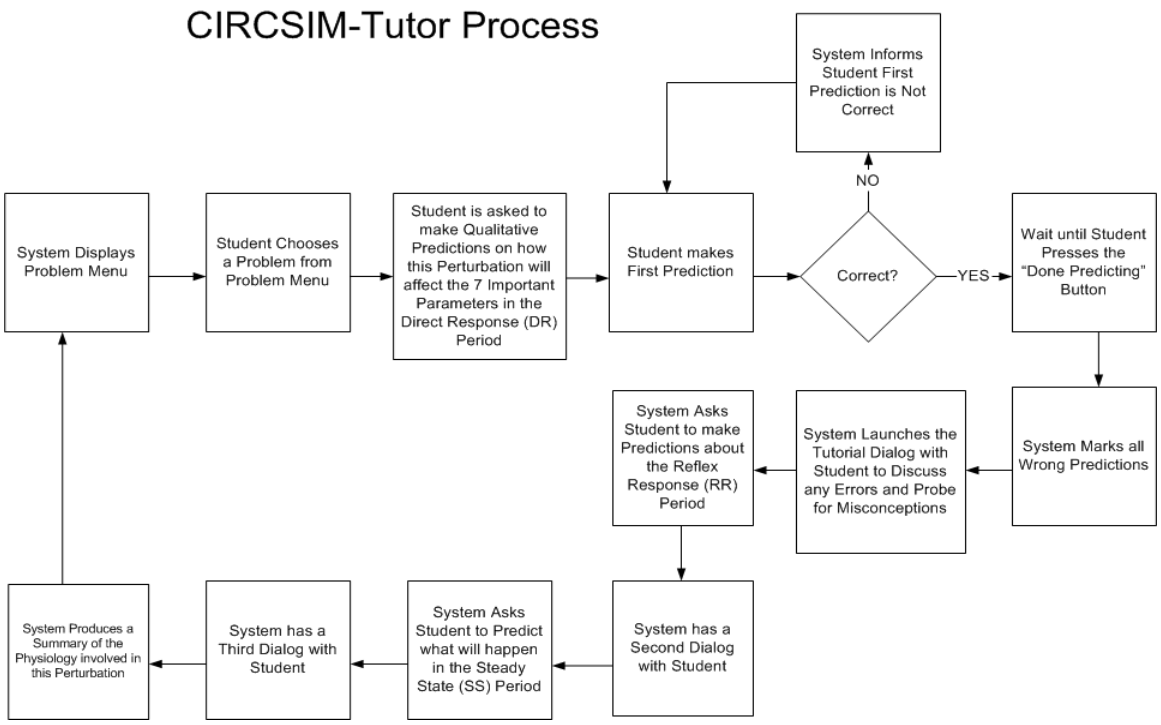

Figure 4: The Tutoring Process in CIRCSIM-Tutor

the session grows as the session develops. A network of MicroDroids is developed each bringing with its own piece of the session ontology. As the session grows, the number of MicroDroids needed will increase. Therefore, more and more experts (Qualitative Reasoners) are instantiated to help solve the problem.

Conclusion

If you want share knowledge and the ontologies for that knowledge, the maintenance can become overwhelming if you use general agents with an ontology server or ontology agent concept. If you really want to share knowledge between systems, the agents need to be more fine-tuned and task-specific. The ontologies for these task-specific agents need to be part of the agent with their own ontology inference engine that does Qualitative Reasoning. This is the concept behind MicroDroids. With this concept, as the end user session grows, the number of MicroDroids used also grows. The session ontology then grows dynamically as the number of MicroDroids that are needed increases. This means that the Qualitative Reasoning capabilities are also expanded as the new MicroDroids are added to solve specific problems.

The concept of embedding models (ontologies) into tutoring systems is not a new one (Bredeweg and Forbus 2003). Better standards are needed. Using UML models that can generate Java objects is a good start.

TKC is not only a collection of knowledge managed by the MicroDroids, but it is also a collection of ontologies modeling the knowledge. This makes TKC an intelligent reusable knowledge base for doing Qualitative Reasoning.

Acknowledgments

I would like to thank ILOG® for supplying their JRules™ System for this work.

This work was partially supported by the Cognitive Science Program, Office of Naval Research under Grant 00014-00-1-0660 to Stanford University as well as Grants No. N00014-94-1-0338 and N00014-02-1-0442 to Illinois Institute of Technology. The content does not reflect the position or policy of the government and no official endorsement should be inferred.

References

Bredeweg, B. and Forbus, K. D. (2003). Qualitative Modeling in Education. *AI Magazine*, Volume 24, No. 4, pages 35-46.

Budinsky, F., Steinberg, D., Merks, E., Ellersick, R. and Grose, T. (2004). *Eclipse Modeling Framework.* Reading, MA: Addison-Wesley.

Evens, M. and Michael, J.A. (In Progress). *One on One Tutoring by Humans and Computers.* Hillsdale, NJ: Lawrence Erlbaum.

Falkenhainer, B. and Forbus, K. D. (1988). Setting up Large-Scale Qualitative Models. In *Proceedings of the American Association for Artificial Intelligence* (AAAI-90). St. Paul, MN. Pp. 301-301.

Fillmore C.J. and Baker, C.F. (2001). Frame Semantics for Text Understanding. In *Proceedings of WordNet and Other Lexical Resources Workshop,* NAACL, Pittsburgh, PA, June, 2001.

FIPA (2002a). FIPA Abstract Architecture Specification. Foundation for Intelligent Physical Agents. Geneva, Switzerland.

FIPA (2002b). FIPA Ontology Service Specification. Foundation for Intelligent Physical Agents. Geneva, Switzerland.

Forbus, K. D. (1985). Qualitative Process Theory. In *Qualitative Reasoning about Physical Systems.* Cambridge, MA: The MIT Press. pp. 85–168.

Forbus, K. D. (1996). Qualitative Reasoning. In *CRC Handbook of Computer Science*, Boca Raton, FL: CRC Press.

Frankel, D. S. (2003). *Model Driven Architecture: Applying MDA to Enterprise Computing.* Indianapolis, IN: Wiley Publishing, Inc.

Franklin, S., and Graesser, A. (1996). Is It an Agent or Just a Program? A Taxonomy for Autonomous Agents. In *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages.* New York: Springer-Verlag. pp. 21-35.

Gallardo, D. and Burnett, E., McGovern, R. (2003). *Eclipse in Action: A Guide for Java Developers.* Greenwich, CT: Manning Publishing Co.

Hutcheson, D. S. (2003). Architecture Comes Alive for IBM. In *Enterprise Architect.* Vol. 1 No. 2. Fawcette Technical Publications Inc. Palo Alto, CA. pp. 41-45.

Michael, J.A., Rovick, A.A., Glass, M.S., Zhou, Y., and Evens, M. (2003). Learning from a Computer Tutor with Natural Language Capabilities. *Interactive Learning Environments, 11(3),* 233-262. Nov. 2003.

Yusko, J. A. (1984). FBL: Frame Building Language. Final Project CSC580, Dept. of Computer Science. Chicago, IL: DePaul University.

Yusko, J. A. (1994). The Reality of Change. Internal technical paper. Unlimited Solutions, Inc. Lombard, IL.

Yusko, J. (In Progress). The Knowledge Collective: A Multi-Layer, Multi-Agent Framework for Information Reuse in an Intelligent Knowledge Base. Ph.D. Thesis, Dept. of CS, IIT. Chicago, IL.

Yusko, J. A. and Evens, M. (2002). The Knowledge Collective: Using MicroDroids to Turn Meta Data Into Meta Knowledge. In *Proceedings of the Thirteenth Midwest Artificial Intelligence and Cognitive Science Conference.* Chicago, IL. pp. 56-60.