Toward Distributed Use of Large-Scale Ontologies[†]

Bill Swartout
Ramesh Patil
Kevin Knight
Tom Russ

USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292
{swartout, ramesh, knight, tar}@isi.edu

Abstract

Large scale knowledge bases systems are difficult and expensive to construct. If we could share knowledge across systems, costs would be reduced. However, because knowledge bases are typically constructed from scratch, each with their own idiosyncratic structure, sharing is difficult. Recent research has focused on the use of ontologies to promote sharing. An ontology is a hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a knowledge base. If two knowledge bases are built on a common ontology, knowledge can be more readily shared, since they share a common underlying structure. This paper outlines a set of desiderata for ontologies, and then describes how we have used a large-scale (50,000+ concept) ontology to develop a specialized, domain-specific ontology semi-automatically. We then discuss the relation between ontologies and the process of developing a system, arguing that to be useful, an ontology needs to be created as a "living document", whose development is tightly integrated with the system's. We conclude with a discussion of Web-based ontology tools we are developing to support this approach.

Introduction

Current knowledge bases are difficult to share or re-use, even when they are expressed in the same formalism and cover the same domain. In our view, this problem stems from the lack of a shared terminology and structure for the knowledge bases. In building a knowledge base, there are many intermediate concepts that a system builder must create and organize to get from specific domain level terms and the very high level concepts that a knowledge representation system provides by default (like "THING"). The decisions about just what those intermediate concepts should be, and how they should be structured, may seem to be somewhat arbitrary since there are many possible organizations that can work.

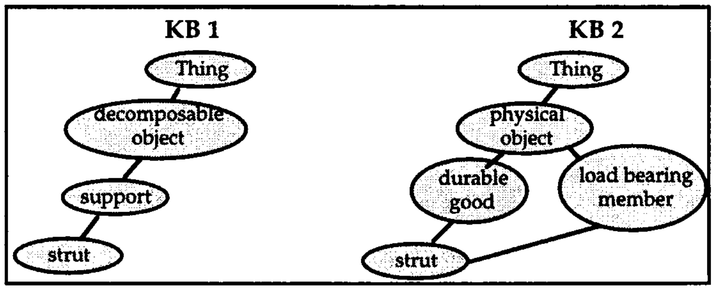


Figure 1: Two ways to represent a "strut"

For example, consider the two knowledge base fragments shown in Figure 1. Both of these fragments represent the concept of a "strut" such as might be part of an aircraft landing gear assembly, but the intermediate concepts used are completely different. KB1 reflects an orientation based on the semantics of natural language, while KB2 uses commonly occurring domain terms. This difference in structure and terminology makes it difficult to share or combine these two knowledge base fragments.

We as well as others [Neches et al 1991; Gruber 1993; Swartout et al 1993] have suggested that basing a representation on an ontology or set of ontologies might provide the answer. An ontology is a set of structured terms that describes some domain or topic. The idea is that an ontology provides a skeletal structure for a knowledge base. If two system builders build their knowledge bases on a common ontology, the systems will share a common structure, and it will be easier to merge and share the knowledge bases.

Although the use of ontologies suggests a possible approach to building sharable knowledge bases, it also raises a number of questions:

- Where does an ontology come from?

[†] A longer version of this paper appeared in the Proceedings of the 1996 Banff Knowledge Acquisition Workshop.

- How should it be organized?

- What tools are needed to support its use in distributed, collaborative efforts?

In this paper, we discuss our experiences in using a large (50,000+ concept) ontology and the tools we are developing to support the use of ontologies during system development. We begin with a discussion of some desiderata for ontologies.

Desiderata for Ontologies

We begin this discussion by drawing a distinction between two different types of ontologies. *Domain ontologies* provide a set of terms for describing some domain, such as medicine, air campaign planning, or computer maintenance. Domain ontologies can be very large and include thousands of concepts. *Theory ontologies,* on the other hand, provide a set of concepts for representing some aspect of the world, such as time, space, causality, or plans. Theory ontologies tend to be more abstract and smaller than domain ontologies. Domain ontologies can be thought of as providing a taxonomy of the relevant objects in some domain, while theory ontologies specify an approach to representation of some aspect. Most of our work concerns domain ontologies, while others (see [Gruber 1992; Tate 1996]) have investigated theory ontologies. Our desiderata apply primarily to domain ontologies.

To be useful throughout the lifecycle of a system, there are a number of characteristics that an ontology should have:

1. *An ontology should not "over commit" on representational choices.* A common error in building a knowledge based system is to attempt to design in detail the various elements of the knowledge base, such as the roles and relations on concepts, constraints among them and so forth, *before* considering how (or if) those elements would be used in problem solving. In the best case, this just results in wasted effort, since some of the structure that is created turns out to be irrelevant. In the worst case, major portions of the knowledge base may have to be redone since the structures chosen beforehand will not work for the problem solving that needs to be done. Since an ontology is, in essence, a pre-package design for a knowledge base, it is important that it leave some representational choices open so that they can be made later, based on how knowledge will actually be used in problem solving. In the ontologies we have developed, the subsumption (a-kind-of) relations between concepts are specified, but the relations on concepts (that is, their "slots") is not specified, since the particular kind of problem solving that will be done usually dictates what relations are most appropriate. We conjecture that one of the reasons why KADS [Wielinga and Breuker 1986] has been so widely accepted is that it started out as an informal, "on paper" approach that provided guidance to system builders without over-constricting their design choices (as might have happened in a more formal method).

2. *An ontology should be extensible.* Because it is difficult to envision in advance how an ontology might be used, it should be possible to extend the ontology to cover new areas as they arise. Extension should be possible both at a low level, by adding domain-specific subconcepts, or at high level by adding intermediate or upper level concepts that cover new areas.

3. *The ontology should be extended based on needs identified during actual use.* In general, there are several possible ways that an ontology might be extended. For example, to capture a causal relation one could choose to represent it as a relation between two concepts, or one could reify the relation as a concept. How should a system builder choose among them? These seemingly arbitrary choices are a problem, particularly if the ontology is developed collaboratively by more than one individual, since if the collaborators make different choices the knowledge base will not be consistent. On the other hand, the choice becomes clearer, if one takes into account the kinds of reasoning that the system must support and the problem solving methods that provide that reasoning, since some of the representational choices will not support the required reasoning. Thus, if extension is guided by real use, it constrains the possible extensions that might be made, and helps ensure that the extensions are consistent and coherent.

4. *Ontologies should not be "stovepipes."* The derisive term "stovepipe system" is used to describe a system that may be vertically integrated but cannot be integrated horizontally with other systems. We want a methodology for building ontologies that will allow us to integrate two independently developed ontologies, even if the need to integrate the ontologies was not anticipated in advance.

5. *An organizing principle (or principles) should be used to structure an ontology.* In building an ontology (or any knowledge base) many of the design decisions, such as what concepts to use, may seem arbitrary. Indeed, without some guiding principles, decisions may be made inconsistently in different parts of the ontology, reducing its coherence. In our ontology work, we have used linguistics as a guide for identifying the concepts that should be in the ontology (particularly the upper level concepts). Linguistics is not the only guidance that could be used. It might be possible, for example, to use conceptual clustering [Michalski 1980] as a guide for ontology structure.

We have used these desiderata to guide our development of both ontologies and the tools that make them useful in building systems. The next three sections describe an experiment we performed using a large-scale ontology to create a specialized domain ontology.

Approach to Ontology Construction

Our approach to ontology construction has been to start with a broad coverage, skeletal ontology that contains over 50,000 concepts. This ontology includes both high and intermediate level terms, but it generally does not include domain specific terms. We then link domain specific terms to the ontology and extend it as the needs of a particular domain require (see

Figure 2). To reduce storage requirements and increase efficiency, we then prune out irrelevant concepts from the broad coverage ontology to produce a focused ontology that includes domain level terms organized by the upper level terms in the broad coverage ontology.

A major advantage of this approach is that it helps us avoid "stovepipe" ontologies. If we start in one domain such as air campaign planning, the broad coverage ontology provides a pre-existing structural base that will allow us to grow our initial ontology to cover other areas. Alternatively, if two ontologies are developed independently (such as logistics and transportation planning in the figure) the broad coverage ontology can act as a "hinge" that couples the terminology and organization of one ontology with the other.

But where does the broad coverage ontology come from? The next section addresses that issue.

SENSUS: A Broad Coverage Ontology

SENSUS is a natural language based ontology that the Natural Language group at ISI is developing to provide a broad conceptual structure for work in machine translation [Knight and Luk 1994; Knight et al. 1995]. It includes both high-level terms (such as "inanimate object") as well as specific terms (such as "submarine"). The terms are organized into an AKO (subsumption) lattice. Each concept in the lattice corresponds to a word sense. Thus, for the word "strut", SENSUS contains three concepts corresponding to the different meanings of the word "strut". Currently, SENSUS contains well over 50,000 concepts.

In contrast to other broad coverage ontologies, such as the ontology that is part of CYC [Lenat and Guha 1990], SENSUS was developed by extracting and merging information from existing electronic resources, rather than constructed from scratch. We will present a review of the construction process here; the original work is described in [Knight and Luk 1994].

SENSUS provides a hierarchically structured concept base whose terms are used as tokens in interlingua expressions for machine translation. The abstract terms at the top of the

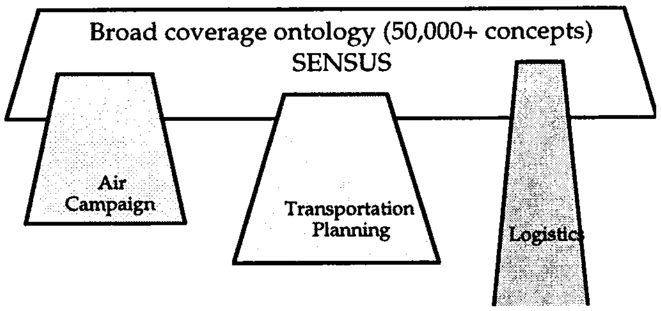


Figure 2: Linking Domain Terms to a Broad Coverage Ontology

hierarchy reflect linguistic generalizations, while the intermediate and lower level terms are familiar, "everyday" terms. Because the machine translation system operates in unrestricted domains, the concept base needs to be very broad.

In building SENSUS, it was found that a number of existing resources had some of the necessary characteristics, but none of them had them all. The PENMAN Upper Model [Bateman et al 1989] and ONTOS [Nirenburg and Defrise 1992] were two very high level, linguistically-based ontologies. They could provide a high level organization, but since each contained only a few hundred concepts, they lacked the necessary broad coverage. WordNet [Miller 1990], a thesaurus-like semantic net, had broad coverage, and was hierarchically organized, but lacked the upper level structure. Electronic natural language dictionaries also had broad coverage and semantic categories that associated certain word senses with particular fields, by identifying, for example, terms from medicine or biology.

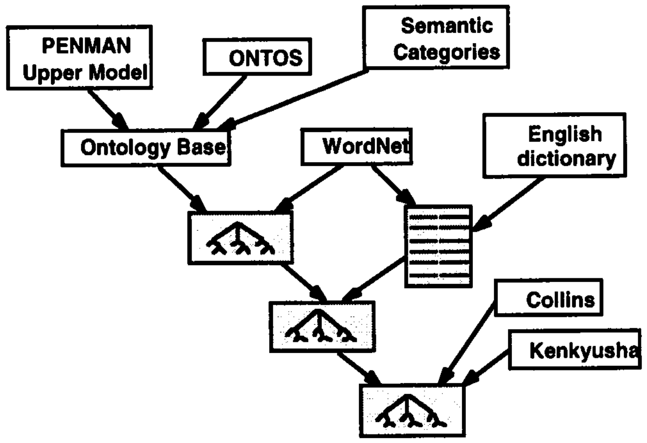


Figure 3: SENSUS Merging Strategy

Since each of these resources had some of the required features, but none had all of them, [Knight and Luk 1994] created SENSUS by merging the resources, so that SENSUS would bring together the needed characteristics. This process began by merging the PENMAN Upper Model, ONTOS and the semantic categories from the

```
3 senses of strut as noun
Sense 1
"strut, swagger" | swagger, strut
  =>"manner of walking" | walk, manner_of_walking
    =>"bearing, carriage" | carriage, bearing
Sense 2
"strut/brace" | strut
  =>"brace, bracing" | brace, bracing
    =>"structural member" | structural_member
      =>"support/supporting structure" | support
        =>"supporting structure" | supporting_structure
          =>"construction, structure" | structure, construction
            =>"instrumentality/artefact" | instrumentality
              =>"artifact" | artifact, article, artefact
                =>"inanimate object" | object, inanimate_object, physical_object, thing
                  =>entity
Sense 3
"prance/gait" | strut, prance, swagger
```

Figure 4: Looking up "strut" in SENSUS

dictionary by hand to produce an ontology base (see Figure 3). WordNet was then merged (again by hand) with the Ontology Base.

A merging tool was then used to merge WordNet with the English dictionary. The tool is semi-automatic in the sense that it looks for corresponding terms in two ontologies and proposes them to a user for confirmation. The reason why it is difficult to find correspondences is that many English words have multiple senses. The tool must try to figure out whether "bank - sense 1" in one ontology corresponds to "bank - sense 3" or "bank - sense 5" in the other ontology. The merging tool uses two techniques to try to identify matches. The first tries to find matches by looking for similarities in the textual definitions that are associated with the concepts in each ontology. The second technique, which is more reliable, uses the hierarchical structure of the ontologies. It begins by identifying the correspondence between unambiguous terms in both ontologies (such as "sandpiper"). The tool then moves up and down the hierarchy from these fixed points. If it finds ambiguous terms that are in structurally similar positions it proposes them as possible correspondences. For example, if "bird - sense 1" in one ontology and "bird - sense 5" in the other ontology are both parents of the term "sandpiper" in their respective ontologies, the tools proposes that the two senses of bird are in correspondence. To support machine translation, the result of this merge was then augmented by Spanish and Japanese lexical entries from the Collins Spanish/English dictionary and the Kenkyusha Japanese/English dictionary.

SENSUS has provided the lexicon and interlingua needed for ISI's Japanese/English machine translation system [Knight et al. 1995]. Further, it helped coordinate the development of a Spanish/English translation system, which was a distributed effort, involving three different research groups at ISI, CMU and NMSU working together. SENSUS helped standardize the terminology and representations that were used among the three sites.

Thus, SENSUS is valuable as an ontology for machine translation. But with over 50,000 concepts, SENSUS also contained many of the concepts that one might need in building a knowledge based system. An interesting possibility occurred to us: Could SENSUS provide the basis for an ontology for a knowledge based system? Informal initial examinations of SENSUS looked promising. For example, looking up "strut" in SENSUS produced the results shown in Figure 4

Sense 2 of "strut" is clearly the sense we had in mind in the example of Figure 1. It is interesting to note that the structure above "strut" in SENSUS is substantially more extensive than in either example in Figure 1. While those examples are admittedly strawmen, we argue that they are not atypical of the sorts of structures one finds in many knowledge based systems. It seemed that a larger test of SENSUS as an ontology for knowledge based systems might be revealing.

SENSUS as a basis for a domain-specific ontology

As a test, we decided to use SENSUS to construct an ontology for military air campaign planning. We decided that it would not be practical to start with SENSUS and just add domain specific terms to it as needed. A 50,000+ concept ontology like SENSUS is somewhat unwieldy, and storing and manipulating it consumes considerable computing resources. If we were to deliver an ontology that included all of SENSUS *and* the domain specific terms as well, we felt that system builders would not be enthusiastic about the performance cost, particularly if most of the concepts in the ontology are irrelevant to their concerns, as will be the case for most domains. To make SENSUS usable as a basis for a domain-specific ontology, we needed a way to identify the terms in SENSUS that were relevant to a particular domain, and then prune the ontology so that it included only those terms.

We began with approximately 60 "seed" terms that domain experts in air campaign planning identified for us. These seed terms were linked by hand to SENSUS, as shown in Figure 5, where the seed terms are cross-hatched. (These figures are intended to be illustrative. The number of concepts actually involved was far greater than can easily be shown graphically.) We included all the concepts on the path from the seed terms to the root of SENSUS, as shown in Figure 6. The semantic categories derived from the English dictionary included a category for "military"

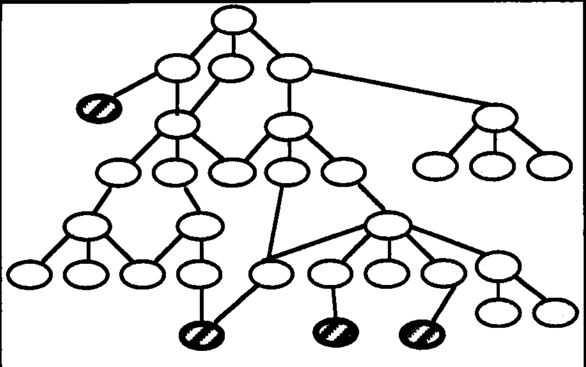


Figure 5: Linking domain terms to SENSUS

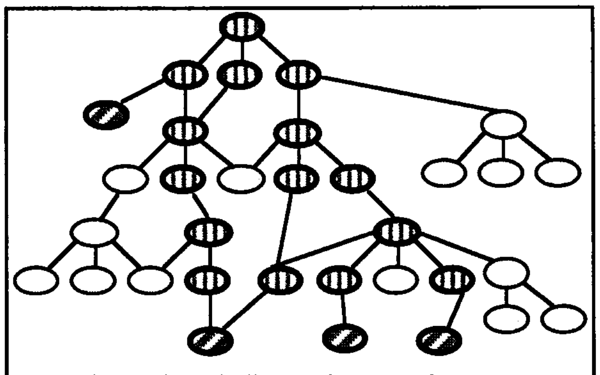


Figure 6: Including nodes on path to root

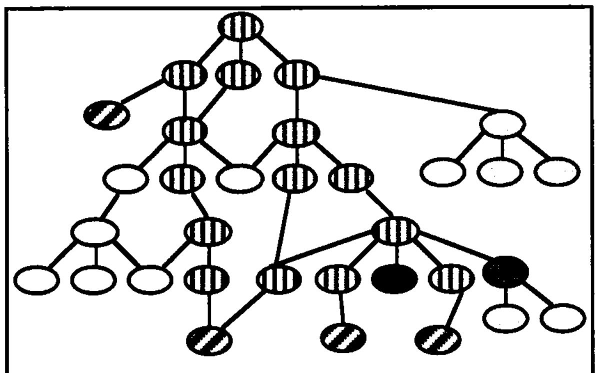


Figure 7: Adding "military" terms

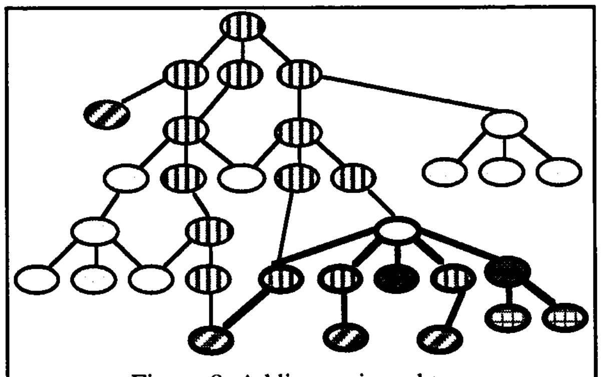
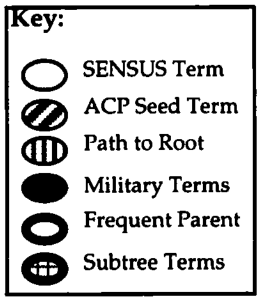


Figure 8: Adding entire subtree



terms. Since air campaign planning is a military domain, we included all the military terms. These are shown in dark gray in Figure 7. Finally, for those nodes that had a large number of paths through them we decided in some cases to include the entire subtree under the node (see Figure 8), based on the idea that if many of the nodes in a subtree had been found to be relevant, then the other nodes in the subtree were likely to be relevant as well. This step was done manually, since it seemed to require some understanding of the domain to make the decision. (Note that very high level nodes in the ontology will always have many paths through them, but it is almost never appropriate to include the entire subtrees under these nodes.)

Air Campaign Ontology Results

Starting with roughly 60 seed terms, we used the process outlined above to generate an ontology. The resulting ontology contained approximately 1600 concepts. This included high-level organizational terms, such as "inanimate object" as well as domain-specific terms that were *not* in the original seed terms, such as: aircraft carrier, ammunition, and battle group. The entire process of generating the ontology from the seed terms took substantially less than one person-week of effort.

As an example, some fragments of the knowledge base showing high-level terms in the ontology, location/spatial terms, and military structure terms are contained in Figure 9 through Figure 10. To us, one of the most surprising things about this exercise was the range of relevant concepts that appeared in the final ontology, even though they were not part of the initial seed set. On the other hand, it should be understood that the pruning mechanism we used was heuristic, and some concepts that were relevant in only a humorous way managed to sneak in, such as "Col. Blimp" (who crept in as a military person) and "pirate ship" (which got in when we included the subtree under ship).

```
: : : OBJECT
: : : : CONCEPT_0003
: : : : : CONSCIOUS-BEING
: : : : : : ANIMAL
: : : : : : : MALE-ANIMAL-OR-FEMALE-ANIMAL
: : : : : : : : FEMALE-ANIMAL
: : : : : : : : : female person...*
: : : : : : : : : female/animal...*
: : : : : : : : MALE PERSON-OR-FEMALE PERSON...*
: : : : : : : : MALE-ANIMAL
: : : : : : : : : male person...*
: : : : : : : : : male/animal...*
: : : : : : : animate being...*
: : : : : : : someone...*
: : : : : NON-CONSCIOUS-THING
: : : : : : SPATIAL-TEMPORAL
: : : : : : : ASPECTUAL
: : : : : : : : DURATION-ASPECT
: : : : : : : : : MOMENTARY
: : : : : : : : : PROLONGED
: : : : : : : : ITERATION-ASPECT
```

Figure 9: High-level terms

```
: : : : : construction, structure
: : : : : : building complex
: : : : : : : establishment/building complex
: : : : : : : : facility/establishment
: : : : : : : : : military installation
: : : : : : : : : : armory, arsenal
: : : : : : : : : : base of operations
: : : : : : : : : : : air station
: : : : : : : : : : : army base
: : : : : : : : : : : firebase
: : : : : : : : : : : navy base
: : : : : : : : : : emplacement/installation
: : : : : : : : : : : gun emplacement
: : : : : : : : : : : : nest/gun emplacement
: : : : : : : : : : : : pillbox/gun emplacement
: : : : : : : : : : military headquarters
```

Figure 10: Military structures Fragment

```
: : : : : : : SPACE-INTERVAL
: : : : : : : : CONCEPT_0058
: : : : : : : : : ONE-OR-TWO-D-LOCATION
: : : : : : : : : : area, region
: : : : : : : : : : : district, territory
: : : : : : : : : : : : administrative division
: : : : : : : : : : : : : province, state
: : : : : : : : : : : geographical area
: : : : : : : : : : : scene/area
: : : : : : : : : : : : THEATER-OF-WAR
: : : : : : : : : : line/location
: : : : : : : : : : surface/location
: : : : : : : : : THREE-D-LOCATION
: : : : : : : SPACE-POINT
: : : : : : : point/location
: : : : : : : : mathematical point
: : : : : : : : : midpoint
: : : : : : : : : : centre of gravity
: : : : : : : : : : : CENTER-OF-GRAVITY
: : : : : : : : topographic point
: : : : : : : : : target area
```

Figure 11: Location/Spatial Terms Fragment

In this section, we've shown how a large-scale ontology can help in rapidly creating an extensible domain-specific ontology. However, creating the ontology is only part of the story. For an ontology to actually be used, tools are needed that integrate the ontology into a system's development process. In the next section, we discuss our vision for how ontologies can be used in system construction and some of the tools we have constructed to support that vision.

Toward Usable Ontologies

Recent research initiatives, such as the DARPA/Rome Planning Initiative (ARPI) [Fowler et al, 1995], have involved teams of researchers working in a common domain with the goal of integrating their systems to produce a much larger and more capable system than any group could produce alone. This approach is a contrast to earlier efforts in which researchers tended to work largely independently. Within ARPI, several ontologies have been developed. The hope was that these ontologies would serve three important roles. First, they could help researchers become familiar with a domain by browsing the ontology to learn the terminology and some of the semantics of the domain. This could reduce the amount of time that was spent with domain experts, who are generally a scarce resource. Second, the ontology could speed system development, by providing system builders with a large base to build upon. Third, when it came time to integrate systems, the ontology could help by providing common terminology to support inter-system communication.

Although the ontologies developed within the ARPI effort have been used somewhat, it is fair to say that they have not been used widely and the hopes for them have not been realized. Several reasons can be identified for this problem:

- *Lack of early ontology support.* Generally, ontology development efforts and the research projects started at about the same time. As a result, developers will made up their own knowledge-bases in order to proceed with their individual projects. Thus, even when a satisfactory common ontology was eventually developed, reconciling the differences between early ontological commitment made by individual researcher with the common ontology was, in many cases, too much effort.
- *Lack of browsing tools:* Early ontologies were often distributed as source code in a particular representation language. Only the support tools present in the KR system used for encoding the ontologies were available. As a result, it was very difficult for system developers to understand the ontology. We believe, that support tools for browsing ontologies (designed specifically with large ontologies in mind) are essential, particularly for the domain familiarization role of ontologies. Users should be able to browse

through an ontology in order to understand its scope, structure and content. They should be able to rapidly search the ontology for terms of interest and related terms, much in the fashion of on-line dictionaries and thesauruses. Only when the users are familiar with the contents and are able to locate the terms of immediate need, will an ontology be used as the terminological foundation for other systems.

- *Lack of translators:* Ontologies were delivered in some particular representation language, but system developers used a variety of knowledge-representation systems and programming languages to support their individual needs. In the previous efforts, it was left up to individual developers to translate and incorporate the ontologies within their systems. This required significant up-front effort on the part of the developers (if it was done at all). To overcome this we need tools for specifying and extracting subset of ontology relevant to a given application, and translators that can translate descriptions out of ontologies in formats suitable for system developers (e.g., Ontolingua, C++ and IDL).
- *A one-shot or release-based approach to ontology development.* Rather than growing with the systems in a tightly-coupled fashion, ontologies were either released once and not extended, or versions of an ontology were made available at periodic intervals. If a particular system builder wanted to extend the ontology, it took a long time for his extension to be reflected in the shared ontology. Furthermore, when a new version of an ontology was released, developers were reluctant to take on the burden of updating their software to track changes in the ontology. As a result, the ontology and the implementations diverged, thus negating the advantages of shared ontology.

The problems above all stem from the failure to integrate ontology use and development directly into the process of system development itself. That is, rather than regarding the ontology as a separate resource that is updated periodically, the development and extension of the ontology should occur as part of system development. In this way, the ontology becomes a "living document" that is developed collaboratively and, at any given time, reflects the terminology that is shared by developers within an effort. Thus, if a system builder needs to extend the ontology, he should be able to do so, perhaps on a private copy of the ontology initially, and then make his augmentations available to others by integrating his changes into the shared ontology.

To support this collaborative view of integrated system and ontology development, it is important that the ontology, and the ontology development environment itself, support group activity, such as, simultaneous viewing, editing and updates. Towards this end, the ontology server must support, version control, ability to check in and out modules for local updates. Additional tools are needed that can guide in the task of adding new knowledge, verifying that the new knowledge is coherent and consistent with the existing base, and in tracking the evolution of ontology by highlighting differences between ontologies (e.g., identify concepts that have been added, deleted or modified) and differences between concept definitions.

The vision of distributed collaborative ontology development was initially explored by the Ontolingua group at Stanford [Farquhar et al. 1995] under the Knowledge Sharing Effort [Neches et.al. 1991]. A Web Based ontology browsing and editing environment for Ontolingua [Gruber 1992] has been in operation since January 1995 and has been received considerable attention within the knowledge sharing community[Rice et al. 1996; Farquhar et al. 1996]. In addition, tools have been developed to translate Ontolingua into and out of other languages such as Loom and IDL. The Ontolingua system has been used to develop sharable foundation ontologies in many areas, such as abstract algebra, units and dimension, component-connection models, and configuration problem solving [Gruber 1993].

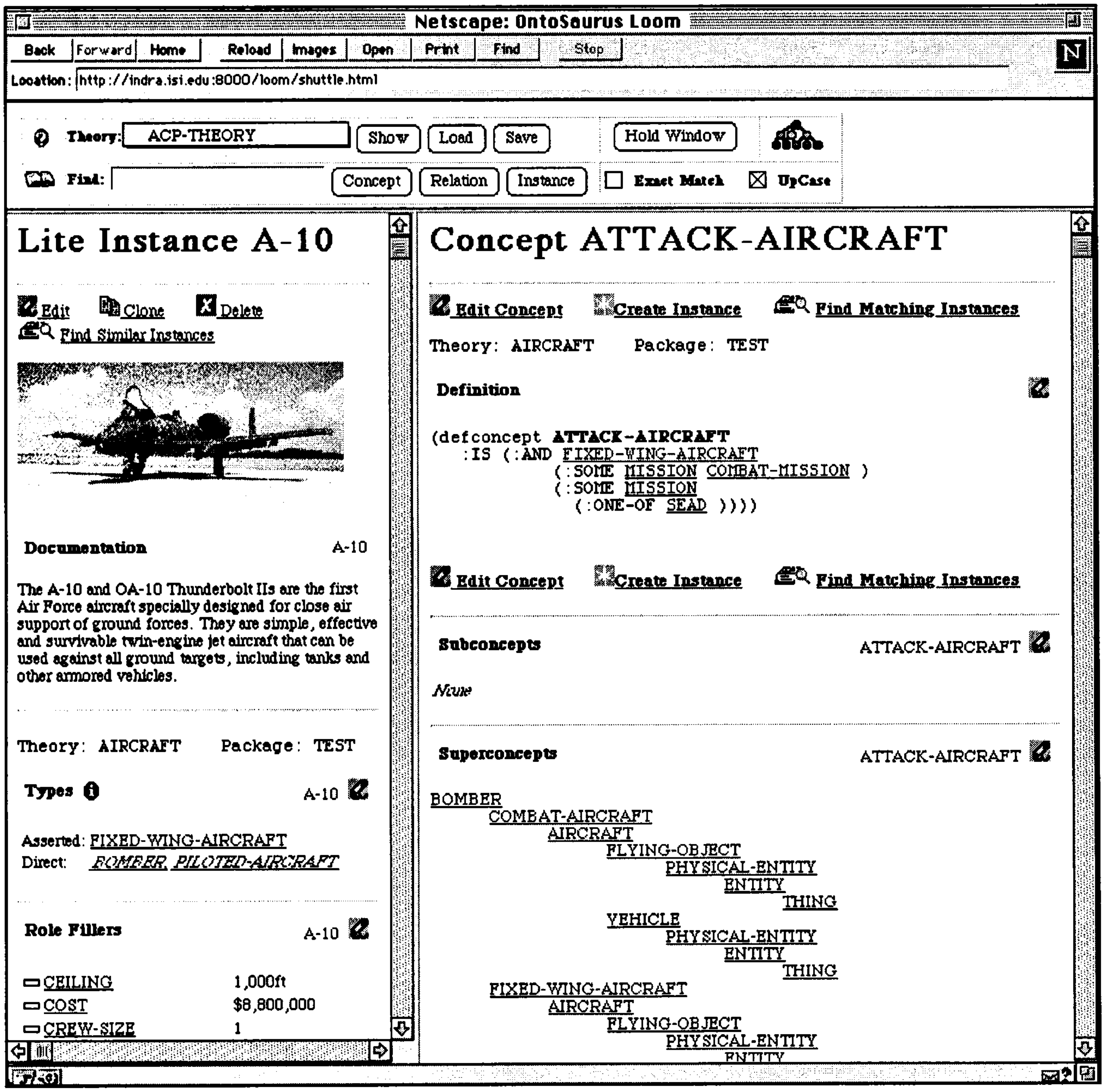
A key difference in the approach we are advocating is the use of broad coverage general ontologies as a starting point for constructing domain-specific ontologies. Our experience with SENSUS ontology and the knowledge sharing ontology library [Gruber 199] suggests that careful development of foundational ontologies for high level domain concepts such as the time, space, and mathematical concepts must be complemented by broad coverage ontologies that address everyday concepts and vocabulary of a domain. These two extremes in a continuum of the ontology development require different considerations: the foundational ontologies tend to be small but require considerable deliberation and analysis in their development whereas the domain ontologies tend to be massive, require considerable domain knowledge and close participation of domain experts in their development. The terminology, the structure and organization of the domain, and a crisp definition for each term in the ontology play central role in domain ontologies. A concept definition in a domain ontology tends to be fairly straightforward. In contrast, defining a concept in a foundational ontology may raise subtle issues that need to be addressed. An additional consideration stems from the respective sizes of fundamental and domain ontologies. Fundamental ontologies tend to be quite small, but domain ontologies may involve thousands of concepts. This distinction changes the need for tools to help maintain the ontology. While a tool for maintaining the concept hierarchy, such as the Loom classifier [MacGregor 1991; 1994] may be helpful for a small fundamental ontology, it is much more critical for maintaining a large domain ontology, since its scale makes it much harder to maintain manually. Thus, the tools and techniques for rapid development of domain ontologies demand a different focus than those for the foundational ontologies.

The Ontosaurus Browser

This section describes our work in constructing a distributed ontology browser and editor. To support our collaborative vision of ontology development, an ontology must be widely available and editable, yet to maintain coherency and consistency, it should be centrally maintained. To us, Web-based technology (HTTP/HTML) seems ideal for achieving such seemingly conflicting demands. The ontology environment consists of two parts: an ontology server, and ontology browser clients.

The Ontology Server (called Ontosaurus) is implemented using CL-HTTP [Mallery 1994] a highly programmable object-oriented Common-Lisp-based Web server, the Loom knowledge representation system [MacGregor 1991a,b], and Lisp code that interfaces Loom to CL-HTTP for browsing, editing, querying, translating, and other Loom knowledge base maintenance functions.

The client is the widely available NetScape 2.0 (or later version) browser. In response to queries from a user, the Ontosaurus server dynamically creates HTML pages that display the ontology hierarchy and it uses HTML forms to allow the user to edit the ontology. By using the Web, and



Lite Instance A-10

Edit Clone Delete
Find Similar Instances

Documentation A-10

The A-10 and OA-10 Thunderbolt IIs are the first Air Force aircraft specially designed for close air support of ground forces. They are simple, effective and survivable twin-engine jet aircraft that can be used against all ground targets, including tanks and other armored vehicles.

Theory: AIRCRAFT Package: TEST

Types A-10

Asserted: FIXED-WING-AIRCRAFT
Direct: *BOMBER, PILOTED-AIRCRAFT*

Role Fillers A-10

CEILING	1,000ft
COST	$8,800,000
CREW-SIZE	1

Concept ATTACK-AIRCRAFT

Edit Concept Create Instance Find Matching Instances

Theory: AIRCRAFT Package: TEST

Definition

```
(defconcept ATTACK-AIRCRAFT
   :IS (:AND FIXED-WING-AIRCRAFT
             (:SOME MISSION COMBAT-MISSION )
             (:SOME MISSION
               (:ONE-OF SEAD ))))
```

Edit Concept Create Instance Find Matching Instances

Subconcepts ATTACK-AIRCRAFT

None

Superconcepts ATTACK-AIRCRAFT

```
BOMBER
      COMBAT-AIRCRAFT
            AIRCRAFT
                  FLYING-OBJECT
                        PHYSICAL-ENTITY
                              ENTITY
                                    THING
                  VEHICLE
                        PHYSICAL-ENTITY
                              ENTITY
                                    THING
      FIXED-WING-AIRCRAFT
            AIRCRAFT
                  FLYING-OBJECT
                        PHYSICAL-ENTITY
                              ENTITY
```

Figure 12: The Ontosaurus Browser

Netscape as our client, the ontology can be examined and edited on many platforms anywhere a network connection is available, yet the ontology itself is centralized in the server.

As shown in Figure 12, the browser is made up of three frames. The top frame consists of a control panel, which provides a variety of menus and buttons. The control panel allows the user to select an ontology from the library ("theory"), display the documentation page for the ontology ("show"), load an ontology, save changes to an ontology, move the contents of the lower two panes ("hold window"), and search for concept, relation, or instances in the ontology. In addition, the graphic icons on the control panel provide ready access to help documents for Ontosaurus, Loom, and the on-line Loom reference manual.

The two panes below are used to display the contents. Whenever a new document is requested, it is displayed in the lower right frame (called the CONTENT frame). If a user wishes to hold a page in view while exploring various links on the page, he may move the content to the lower left frame (called the TOC frame). Furthermore, while editing a concept or entering a new concept, the two frames allows a user to simultaneous access to the editing form as well as the ontology. Thus, the two lower frames provides a nice environment for navigating, exploring or updating the contents of the ontology, while a static top pane provides ready access to the controls without the need for scrolling through pages.

Because we envision that the ontology will be used both for domain familiarization as well as a formal representation for a knowledge based system, we have included in the ontology items like graphics and textual documentation that are not normally part of a formal representation. Thus, in Figure 12, the left pane shows a picture of an A-10 as well as text documentation for it.

Our design philosophy is that each dynamically generated page should be a complete document, that is, it should bring together all the information relevant to an item. For example, the documentation page for a concept contains (i) image and textual documentation associated with the concept, reference links to other related source documents and references used in formalizing the concept, (ii) a concept definition with hyperlinks to other concepts and relations used in the definition, (iii) its super-concepts, sub-concepts, and sibling concepts, (iv) applicable roles (slots), and (v) instances of the concepts in the knowledge base. Furthermore, we use typographical conventions to distinguish between information that is asserted (standard face), and that is inferred by the Loom classifier (italic). For example, in Figure 12, an A-10 is asserted to be a FIXED-WING-AIRCRAFT, but it is inferred to be a *BOMBER* and a *PILOTED-AIRCRAFT.* All objects (concepts, relations, instances) are hyper-linked (appear in link color), whereas data (i.e. constants such as strings, numbers etc.) are not. Finally, we have designed a collection of character height icons to compactly provide additional information such as single-valued vs. multi-valued relations.

If the user wants to edit a concept or instance, he clicks on the edit button. Fields are provided to edit role restrictions and super-concepts. In addition, a text window is provided for more complex Loom forms.

Because the Ontosaurus server incorporates Loom, it can take full advantage of Loom's reasoning capabilities. In particular, when a new concept is added or an existing concept modified, Loom is used to classify the new description in its proper place in the ontology and to verify that the new description is coherent, and the results of the classification become available to all users of the system.

Our current implementation of the browser thus provides capabilities for browsing, and editing of Loom based ontologies. Translators from Loom to Ontolingua [Gruber 1992], and KIF[Genesereth and Fikes 1992; Genesereth 1991] and KRSS [Patel-Schneider et al. 1993] have been developed that can translate individual Loom objects on demand, and have been incorporated with the browser to allow a user to simultaneously view Loom and its translation during browsing, or to request that a KB be saved in one of these languages for use by other systems. Additionally translation of a Loom knowledge base to a C++ object schema has also been implemented to allow users to generate appropriate C++ ".h" and ".C" files that can be compiled to create C++ object classes and instances corresponding to concepts and instances in the Loom KB.

We are currently working on providing tools for comparing and contrasting different versions of an ontology as it evolves. In addition, we are extending Loom's built-in ontology to support on-line collaboration, such as an "edit history" and documentation of issues. Additional tools for semi-automatic merging of ontologies, and extracting subsets from the ontology (as mentioned earlier) are also planned.

Related Work

The development of Ontosaurus was inspired by the Stanford Ontology sever {http://www.ksl.stanford.edu/} based on Ontolingua. Like the Ontolingua browser, Ontosaurus provides full access to ontologies, allows on-line editing, and provides various translation and ontology management tools. Also like the Stanford Ontolingua server, it is intended to provide an environment for distributed collaborative development and maintenance of ontologies. One difference is that Ontosaurus is intended to help people become familiar with a domain and thus the browser provides support for documentation and graphics to help people become acquainted with domain objects. A more significant difference is that Ontosaurus is built on Loom, a fully functional KR system. Using Loom as an integral part of the ontology development environment allows us to provide the range of reasoning capabilities associated with Loom and to leverage them for providing ontology maintenance services such as the automatic detection of incoherence, inconsistency, and missing definitions.

The ontology development approaches taken in Ontolingua and Ontosaurus also differ significantly. The Ontolingua approach is to construct a collection of relatively small content rich theory ontologies, and to compose them together to form larger ontological structures. The approach taken in Ontosaurus is to develop a large comprehensive linguistically motivated ontology that provides a common organizational framework for the development of a knowledge base. This structure is then enriched with domain terms and facts to generate rich ontologies. We believe that providing a carefully organized overarching framework will greatly simply the task of integrating independently developed systems, and facilitate communication between systems.

KSSn[Gaines 1994, 1995] is also similar to Ontosaurus. KSSn provides many of the KL-One like inference services present in Loom. In addition to web-based browsing and editing capabilities, KSSn, also provides web-based graphical editing capabilities (using application plug-ins).

A number of other researchers are also exploring knowledge-based collaborative engineering. In particular the SHADE [McGuire et al. 1993] system uses an ontology based approach to facilitate collaboration between product development and integrated manufacturing. Shade provides a rich collection of tools for tracking and propagating design changes and allowing distributed teams of engineers to resolve changes to the overall design. The emphasis in SHADE is to support distributed collaboration environment for computer aided engineering. The focus of Ontosaurus is to provide similar support to knowledge-based system engineering.

Summary

Much work remains to be done before the use of ontologies is commonplace. Nevertheless, we can already begin to see a number of benefits that may come from the use of ontologies to support system construction. First, it represents a new paradigm for model construction, where the focus is on linking domain specific terms to an existing ontology and extending it, rather than on building a model from scratch. We argue that this is much easier to do. Second, we think this approach can provide a kind of guidance for representation. The organization of the terms in a large-scale ontology like SENSUS can help organize the domain specific terms that a system builder may attach. Further, some of the distinctions SENSUS draws between word senses may force a system builder to think more carefully about what he is representing, and whether or not he is drawing all the appropriate distinctions. Third, the few "seed" terms that the domain expert provides can identify entire subtrees of the model that are relevant to include. Thus, our approach acts as a kind of concept multiplier, which can greatly speed up model construction. Forth, our approach promotes sharability of models. This could occur within some domain, or by using the ontology base as common underpinning, even models developed in different domains will be easier to share. Fifth, our use of a natural language based ontology means that much of the information that is needed to generate paraphrases and explanations automatically is already represented in the ontology base. This information would have to be represented explicitly otherwise. By easing the production of paraphrases and explanations understandability can be enhanced.

ACKNOWLEDGMENTS

The authors would like to thank Richard Whitney for his insights and help with programming and Eduard Hovy and Lou Hoebel for comments, suggestions and valuable discussions. All of them contributed greatly to this research.

The work described here has been supported by the Defense Advanced Research Projects Agency under contracts DABT63-91-C-0025, DABT63-95-C-0059 and NAG2-581.

REFERENCES

[Abbrett and Burstein, 1987] G. Abbrett and Mark Burstein. The KREME knowledge editing environment. In International Journal of Man-Machine Studies (1987) 27, pp. 103-126.

[Bateman et. al. 1989] J. A. Bateman, R.T. Kasper, J.D. Moore, and R.A. Whitney. A General Organization of Knowledge for Natural Language Processing: The Penman Upper Model. Unpublished research report, USC/Information Sciences Institute, Marina del Rey. 1989.

[Farquhar et al. 1995] A. Farquhar, R. Fikes, W. Pratt, and J. Rice. "Collaborative Ontology Construction for Information Integration" Knowledge Systems Laboratory, Department of Computer Science, Stanford University, KSL-95-63, August 1995.

[Farquhar et al. 1996] Farquhar, A., R. Fikes, J. Rice. The Ontolingua Server: a Tool for Collaborative Ontology Construction. In KAW96. November 1996. Also available as KSL-TR-96-26.

[Fowler, Cross and Owens 1995] N. Fowler III, S. E. Cross and C. Owens. The ARPA-Rome Knowledge Based Planning and Scheduling Initiative, *IEEE Expert,* 10(1) 4-9. February 1995.

[Gaines 1994] B.R. Gaines. Class library implementation of an open architecture knowledge support system. *International Journal of Human-Computer Studies* 41(1/2) 59-107, 1994.

[Gaines and Shaw 1995] B. R. Gaines and M. L. G. Shaw. WebMap Concept Mapping on the Web. In the *Proceedings of the Fourth International World Wide Web Conference,* Boston, Massachusetts. 1995.

[Genesereth 1991] M. R. Genesereth. Knowledge Interchange Format. Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference, Cambridge, MA, pages 599-600. Morgan Kaufmann, 1991.

[Genesereth and Fikes 1992] M. R. Genesereth, R. E. Fikes (Editors). Knowledge Interchange Format, Version 3.0 Reference Manual. Computer Science Department, Stanford University, Technical Report Logic-92-1, June 1992.

[Gruber 1993] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. In *Formal Ontology in Conceptual Analysis and Knowledge Representation*, N. Guarino and R. Poli, editors, Kluwer Academic, in preparation. Original paper presented at the International Workshop on Formal Ontology, March 1993. Available as Stanford Knowledge Systems Laboratory Report KSL-93-04.

[Gruber and Tenenbaum] T. R. Gruber, J. M. Tenenbaum, and J. C. Weber. Toward a knowledge medium for collaborative product development. *Proceedings of the Second International Conference on Artificial Intelligence in Design*, Pittsburgh, pages 413-432. Kluwer Academic, 1992.

[Gruber 1992] T. R. Gruber. Ontolingua: A mechanism to support portable ontologies. Stanford University, Knowledge Systems Laboratory, Technical Report KSL-91-66, March 1992.

[Hatzivassiloglou and Knight 1995] V. Hatzivassiloglou and K. Knight. Unification-Based Glossing. Proceedings of the 14th IJCAI Conference. Montreal, Quebec. 1995.

[Knight and Luk] K. Knight. and S. Luk. Building a Large Knowledge Base for Machine Translation. *Proceedings of the American Association of Artificial Intelligence Conference (AAAI-94).* Seattle, WA. 1994.

[Knight et. al. 1995] K. Knight, I. Chander, M. Haines, V. Hatzivassiloglou, E. H. Hovy, M. Iida, S.K. Luk, R.A. Whitney, and K. Yamada. 1995. Filling Knowledge Gaps in a Broad-Coverage MT System. Proceedings of the 14th IJCAI Conference. Montreal, Quebec.

[Lenat and Guha 1990] D. B. Lenat and R. V. Guha. Building Large Knowledge-Based Systems: Representation and Inference in the CYC Project. Addison-Wesley Publishing Company, Inc., Reading, Massachusetts. 1990.

[MacGregor 94] R. M. MacGregor. A Description Classifier for the Predicate Calculus. in *Proceedings of the Twelfth National Conference on Artificial Intelligence*, (AAAI-94), 1994.

[MacGregor 1991a] R. M. MacGregor. Using a Description Classifier to Enhance Deductive Inference in Proceedings of the Seventh IEEE conference on AI Applications, 1991.

[MacGregor 1991b] R. MacGregor. The Evolving Technology of Classification-Based Representation Systems. In J. Sowa (ed.), *Principles of Semantic Networks: Explorations in the Representation of Knowledge.* Morgan Kaufmann, 1990.

[Mallery 1994] John C. Mallery. A Common LISP Hypermedia Server, in *Proceedings of the First International Conference on The World-Wide Web*, Geneva CERN, May 25, 1994.

[McGuire et. al. 1993] J. G. McGuire, D. R. Kuokka, J. C. Weber, J. M. Tenenbaum, T. R. Gruber, G. R. Olsen. SHADE: Technology for Knowledge-Based Collaborative Engineering. *Journal of Concurrent Engineering: Applications and Research (CERA)*, 1(2), September 1993.

[Michalski 1980] R. S. Michalski. Knowledge Acquisition Through Conceptual Clustering: A Theoretical Framework and an Algorithm for Partitioning Data into Conjunctive Concepts. *Policy Analysis and Information Systems,* 4(3) 219-244, 1980.

[Miller 1990] G. Miller. WordNet: An on-line lexical database. *International Journal of Lexicography* , 3(4). (special Issue). 1990.

[Neches et. al. 1991] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, & W. R. Swartout. Enabling technology for knowledge sharing. Enabling technology for knowledge sharing. AI Magazine, 12(3):16-36, 1991.

[Patel-Schnider et. al. 1993] DRAFT of the specification for Description Logics, produced by the KRSS working group of the DARPA Knowledge Sharing Effort. updated July 1993.

[Rice et al 1996] J. Rice, A. Farquhar, P. Piernot, & T. Gruber. Lessons Learned Using the Web as an Application Interface. Knowledge Systems Laboratory, KSL-95-69, September 1995. In CHI96, April 1996.

[Swartout et. al. 1993] W. R. Swartout, R. Neches and R. Patil. Knowledge Sharing: Prospects and Challenges. In *Proceedings of the International Conference on Building and Sharing of Very Large-Scale Knowledge Bases '93,* Tokyo, Japan 1993.

[Tate 1996] A. Tate. Towards a Plan Ontology. Journal of the Italian AI Association (AIIA) January 1996.

[Wielinga and Breuker, 1986] B.J. Wielinga and J.A. Breuker, Models of Expertise, *ECAI* 1986, 497-509.