A Library of Generic Concepts for Composing Knowledge Bases

Ken Barker and Bruce Porter
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712 USA
{kbarker, porter}@cs.utexas.edu

Peter Clark
Knowledge Systems
Boeing Math and Computing Technologies
m/s 7L66, PO Box 3707, Seattle, WA 68124 USA
peter.e.clark@boeing.com

ABSTRACT
Building a knowledge base for a given domain traditionally involves a subject matter expert and a knowledge engineer. One of the goals of our research is to eliminate the knowledge engineer. There are at least two ways to achieve this goal: train domain experts to write axioms (*i.e.*, turn them into knowledge engineers) or create tools that allow users to build knowledge bases without having to write axioms. Our strategy is to create tools that allow users to build knowledge bases through instantiation and assembly of generic knowledge components from a small library.

In many ways, creating such a library is like designing an ontology: What are the most general kinds of events and entities? How are these things related hierarchically? What is their meaning and how is it represented? The pressures of making the library usable by domain experts, however, leads to departures from the traditional ontology design goals of coverage, consensus and elegance. In this paper we describe our *component library*, a hierarchy of reusable, composable, domain-independent knowledge units. The library emphasizes coverage (what is an appropriate set of components for our task), access (how can a domain expert find appropriate components) and semantics (what knowledge and what kind of representation permit useful composition). We have begun building a library on these principles, influenced heavily by linguistic resources. In early evaluations we have put the library into the hands of domain experts (in Biology) having no experience with knowledge bases or knowledge acquisition.

Keywords
knowledge engineering, ontologies, knowledge reuse

INTRODUCTION
The traditional audience for concept taxonomies includes knowledge engineers, ontologists and philosophers. This audience is often interested in ontologies as elegant models capturing a natural division of kinds of things in the universe of discourse. When the intended audience includes experts in particular fields of knowledge who hope to use the ontology to represent abstractions from their fields, the pressures on the design of the ontology shift.

It is a claim of our research [28] that users with no experience in knowledge engineering will be able to represent knowledge from their domain of expertise by instantiating and composing generic components from a small, hierarchical library. Components are coherent collections of axioms that can be given an intuitive label — usually a common English word. The components should be general enough that their axiomatization is relatively uncontroversial. Composition consists of specifying relationships between instantiated components so that additional implications can be computed.

As a guiding principle in building the library we have chosen to restrict both the number of components (to a few hundred) and the size of the language of composition — the relationships between components (currently less than a hundred). Our goal is to achieve coverage through composition rather than through enumeration of a large number of concepts. The small library and simple composition language also have the benefit of being easy to learn for users with no knowledge engineering experience.

This design principle presents two research questions: 1) is such a system easy for users to master? 2) is such a system sufficient to represent sophisticated domain knowledge? We have evidence that the system is indeed usable by domain experts. The quality of the representations created by our domain experts is under review.

In an attempt to make the library more accessible to users unfamiliar with knowledge engineering, we have taken a somewhat different approach to building our ontology: we have taken inspiration from English lexical resources (such as dictionaries, thesauri and English word lists) and Linguistics research. We are certainly not rejecting traditional knowledge engineering approaches, trying instead to reconcile them with language usage. Rather than try to avoid the clash between knowledge base concepts and

English words, we are attempting to make our component library intuitive to users accustomed to expressing knowledge with natural language.

This paper is part of a larger context of ongoing research on knowledge base construction by composition. Elsewhere we have discussed:

- motivations for the approach and algorithms [5, 6, 7]
- a graphical user interface [8]
- a knowledge representation and reasoning system [6]
- question answering and explanation generation [17, 24]

Within that context, this paper provides a brief tour of an early version of our component library to highlight its requirements, construction, contents and applications.

In the following section, we will describe our research project in more detail and the design constraints it places on our component library. We will then expose the contents of the library: what components it contains, what the language for composing components is and how we arrived at these. We will describe the ways in which the user accesses the library and report on some early observations of domain experts using the library.

The component library itself is online and can be browsed at http://www.cs.utexas.edu/users/mfkb/RKF/tree/.

THE PROJECT

A challenge problem for DARPA's Rapid Knowledge Formation (RKF) project [11] is to provide a software environment in which a biologist can build a knowledge base from information found in a textbook on Cell Biology. It must be possible to query the resulting knowledge base to obtain answers to the kinds of questions typically found at the end of a textbook chapter.

Our component library is being used in software (called *SHAKEN*) being developed by SRI, one of the primary contractors on the RKF project [27]. A user of *SHAKEN* builds a knowledge base by taking generic components from the library, instantiating them in a graph and connecting the instantiations to represent such things as static relationships between concepts, temporal and spatial information, event structure and process plans.

Requirements for Library Components

Given the project requirements, it is imperative that the user have a sufficient variety of components (*coverage*), that components that satisfy user expectations can be found easily (*access*) and that components are general enough to be used in a variety of contexts but specific enough to express non-trivial knowledge (*semantics*).

Coverage

There should be components to allow the user to encode a variety of knowledge from any domain. This is not to say that there should be as many components as there are words in a dictionary. Rather, the library should be broad-coverage with components specific enough that a user is willing to make the abstraction from a domain concept. Conversely, the components should not be so specific that the user is handcuffed or does not care enough about the fine distinctions to use the components consistently.

Access

Although knowledge engineers and philosophers are interested in the structure of upper-level ontologies, it is less likely that a biologist describing DNA replication will be interested in learning our hierarchy in order to find components. Furthermore, since we are restricting the library to a small number of components, it is unlikely that there will be an exact match for a concept required by the user. For both these reasons, it is important that the interface help the user to find appropriate components.

Semantics

Our library is not merely a taxonomy of concepts. Each component contains axioms that encode the meaning of the component as well as how the component interacts with other components. These axioms must be general enough that the components are reusable. They must also be written in such a way that they do not clash with the axioms of other components when composed.

In the next sections we will discuss how these criteria, along with previous successful work on broad-coverage intuitive semantic inventories have guided the construction of our library.

RELATED WORK

In theory an ontology could be strong on all dimensions: coverage, access, semantics. In practice, however, an ontology, like most artifacts, is the result of engineering tradeoffs. For example, consider WordNet [21] and Sensus [16]. On one hand, they are as easily accessed as a thesaurus and have very broad coverage — they include the variety of concepts, relations, and modifiers used in everyday text. On the other hand, they provide very shallow semantics. For each English word, these ontologies give its senses along with their definitions, parts of speech, subclasses, superclasses and sibling classes. The definitions are free text (of limited use to computer programs) and the encoded relations are the only semantics.

The ontologies in Ontolingua [14] represent a different point in the space of tradeoffs. These ontologies are very limited coverage (they apply mainly to isolated topics in Engineering), but they have rich semantics. For example, they can be used to compute answers to Engineering problems stated in their vocabulary.

Cyc [10, 18, 19] represents yet another point. Its coverage is arguably as broad as WorldNet's, including many senses for entries in its lexicon. By one account, however, it receives lower scores on semantics and accessibility. Parmar [23] compared the representations of a handful of actions in Cyc and our component library. She found that Cyc often lacks axioms that capture the effects of actions — the representation does not support automated reasoning about change. In terms of accessibility, Parmar measured the time she spent searching the Cyc ontology for entries that correspond to fifteen common actions represented in the component library.[1] On average, she spent over 3.5 minutes finding the Cyc term that most closely matches each action. By her assessment, many of these matches were not close: on a scale from 1 (poor) to 10 (perfect), the average score was less than 6.5.

It is clear, though, that Cyc is an example of a very different approach to coverage than what we propose, making it difficult to compare Cyc and our library. Cyc achieves coverage through enumeration. The semantics of concepts is often encoded in the fine distinctions between specialized subclasses.

Given our small number of components and relations, there is an obvious overlap with work on semantic primitives in Linguistics and Natural Language Processing. Schank's Conceptual Dependency Theory [25] enumerated a very small number of primitive actions and the relations between actions and their participants. A later refinement [26] included relations between pairs of actions. The extremely small number of primitives forces each one to cover many different concepts. The avoidance of names that clash with English words makes the Conceptual Dependency language less intuitive to users.

For our project, rich semantics is the first priority. The semantics of each component is expressed in KM [6], which in turn is defined in first-order logic. KM includes situation calculus — a knowledge representation and reasoning formalism for actions and the changes they cause. For example, the component for ENTER includes KM encodings of these axioms:

- ENTER is a type of MOVE, so instances of ENTER inherit axioms from MOVE, such as: the action changes the location of the object of the MOVE
- before the ENTER, the object is outside some enclosure
- after the ENTER, the object is inside that enclosure and contained by it
- during the ENTER, the object passes through a portal of the enclosure
- if the portal has a covering, it must be OPEN; and unless it is known to be CLOSED, assume that it is OPEN.

We plan to achieve good coverage by encoding a small set of general components for breadth. Depth can then be achieved through specialization and composition of components, without having the user write axioms.

We consider accessibility especially important, given that our users are not knowledge engineers. The library has been designed to allow retrieval of components by means of semantically related search terms (as described below: see *Searching the Library*).

[1] BREAK, CARRY, CREATE, ENTER, EXIT, MAKE-ACCESSIBLE, MAKE-CONTACT, MAKE-INACCESSIBLE, MOVE, RELEASE, REMOVE, REPAIR and TRANSFER.

THE COMPONENT LIBRARY

In deciding what components to encode, we took inspiration from linguistic resources (such as dictionaries and thesauri). Our goal was not to build an online dictionary, but rather a library of components representing concepts that are general and intuitive enough to have obvious labels among common English words.

Furthermore, since domain experts are accustomed to expressing their knowledge with words, having explicit links between our components and dictionaries will help provide access to the library (as described below). These linguistic resources have much to offer:

- They have broad coverage of common terms. Our goal is to have a library of domain-independent, general components. This is where the strengths of general-purpose dictionaries and thesauri lie.
- Lexicographers pay attention to consensus view of the semantics of terms and common usage. Most dictionaries and thesauri are the result of many years of studying how terms are commonly used.
- They often group semantically related words into general semantic categories. These categories may be thought of as the most general concepts.

The Longman Dictionary of Contemporary English (LDOCE) [29] uses a "defining vocabulary" of about 2,000 words. All definitions in the dictionary ground out eventually to the defining vocabulary. WordNet groups semantically similar words into "synsets", which are themselves linked hierarchically. Roget's Thesaurus [20] divides the universe into six classes. Each class is subdivided into multiple sections, themselves subdivided. The thousand leaves in Roget's tree contain semantically related words (not quite synonyms), one of which is chosen as the representative for the group: the *headword*.

Each of these resources (the Longman defining vocabulary, a horizontal slice of the WordNet hierarchy, the Roget headwords) could be used as a list of general concepts, or as inspiration for an original list. None of these would suit our purposes *as-is*: the LDOCE vocabulary is not organized semantically; WordNet has considerably less coverage and fewer relations among non-nouns; Roget is somewhat arbitrary, and obviously influenced by his culture.

Generic Events

The main division in our component library is between *entities* (things that *are*) and *events* (things that *happen*). Events are states and actions. States represent relatively static situations brought about or changed by actions.

Actions

The actions are grouped into fifteen top-level clusters, each having several more specific subclasses (Table 1). The list was developed under consultation of WordNet, the LDOCE defining vocabulary and Roget.

For example, the list of actions was compared to those headwords in Roget's Thesaurus that most naturally describe actions. In Roget, each headword heads several paragraphs; each paragraph contains words of the same part of speech. Although the headwords themselves are all nouns, some of the nouns are nominalizations and represent events more naturally than entities (for example, headwords #161: Production and #264: Motion). For these headwords, the noun paragraphs are often relatively empty, or contain more nominalizations. Their verb paragraphs are the richest. Although there are over one thousand headwords in Roget, our actions are general enough to cover most of the more action-like headwords (with the exception of those having to do with "sentiment and moral power" — an area we have so far ignored).

States

States are relatively temporally stable events. They are coherent collections of axioms that represent situations brought about or changed by actions. Many of our actions are defined in terms of the change in state they cause.

This relationship between actions and states is made explicit in the library: there are actions that put objects into states, actions that take objects out of states and actions whose behavior is affected by objects being in states. For example, the BREAK action puts an object into a BE-BROKEN state. The REPAIR action takes an object in a BE-BROKEN state out of that State. If an object is in a BE-BROKEN state, it may not be the instrument of any of the events for which it is the intended instrument (though it may be instrument of other actions, such as using a broken computer to hold a door open). Other states include BE-RUINED, BE-CLOSED, BE-CONFINED, BE-TOUCHING, BE-ATTACHED-TO, etc.

There are other events that seem to fit somewhere between our actions and states, such as "being in motion". We expect that most of our actions have non-conclusive, durative counterparts (such as MOVING, CREATING, etc.). We are investigating continuous representations of our actions for the purpose of simulation. For now, our actions are all represented as discrete events.

Entities and Roles

To date, we have concentrated on events. We plan to research generic entities in a similar way. Our entity hierarchy is currently a relatively impoverished tree, serving as the root of a number of concepts from our test domain: Cell Biology (just over 500 at the time of writing).

Our preliminary investigation into entities led us to distinguish a separate class of *role concepts*. A role can be thought of as a temporally unstable entity. It is what an entity *is* in the context of some event. For example, PERSON is an entity while EMPLOYEE is a role. A PERSON remains a PERSON independent of the events in which she participates. Conversely, someone is an EMPLOYEE only by virtue of participation in an EMPLOY event.

Our library allows instances of roles to be linked to instances of entities as *adjunct instances* that can be used to capture both the role that an entity plays in an event, and the role it is intended to play (its *purpose*).

In order to determine how common role concepts are, we

Action	*Description*	*Example Subclasses*
ADD	add a part to an entity	--
REMOVE	remove a part from an entity	--
COMMUNICATE*	transfer information	INTERPRET, ENCODE, REPLY
CREATE	bring a new entity into existence	COPY, PRODUCE, PUT-TOGETHER
BREAK	cause an entity to be unable to be used as instrument (for events in which it is the intended instrument)	DESTROY, RUIN, TAKE-APART
REPAIR	"undo" a BREAK	--
MOVE	change the location of an entity	CARRY, ENTER, SLIDE
TRANSFER	change the possessor of an entity	DONATE, LOSE, TAKE
MAKE-CONTACT	make entities touch	ATTACH, COLLIDE
BREAK-CONTACT	make touching entities touch no longer	DETACH, DISPERSE
MAKE-ACCESSIBLE	allow an entity to participate (in various ways) in events	ADMIT, EXPOSE, RELEASE
MAKE-INACCESSIBLE	prevent an entity from participating in events	BLOCK, CONCEAL, CONFINE
PERCEIVE*	discern using senses	IDENTIFY, TOUCH
SHAPE*	change the shape of an entity	FLATTEN, FOLD
ORIENT*	change the orientation of an entity	FACE, ROTATE, TURN

Table 1: The top-level action clusters (actions marked * are under construction in the library)

conducted an experiment with the Collins online dictionary [9]. In that experiment we estimated that as many as 6% of nouns satisfy our criteria for role concepts. Furthermore, the most frequent nouns in the British National Corpus [4] also contain an estimated 6% role concepts. A more detailed discussion of roles and justification for a separate role hierarchy appear in [13].

COMPOSITION

The precoded axioms in library components provide much of the power that allows domain experts to build knowledge bases. Equally important is the ability to connect components in such a way that our knowledge representation system (KM [6]) can draw inferences from the composition beyond the union of the individual axioms of the components.

From the point of view of a user, composition is simply the linking together of library components. From this linking, however, KM is able to draw inferences by way of the knowledge encoded in components:

- *Conditional rules*: many components specify additional axioms that are asserted conditionally, dependent on the kinds of components they are composed with and the kinds of connections between them. For example, if the raw material of a PRODUCE is a SUBSTANCE, then the product is composed of that SUBSTANCE. If the raw materials are OBJECTs, then the product has those OBJECTs as parts.
- *Definitions*: many components specify the sufficient conditions under which KM can automatically reclassify instances. For example, an instance of MOVE whose destination is inside a container is automatically reclassified as an instance of ENTER, allowing KM to apply the axioms of that more specific component.
- *Simulation*: many components include preconditions that must be satisfied for an action to take place and the axioms that get asserted (or retracted) as a result of the action taking place. KM is able to simulate complex combinations of events and their participating entities.

The Language of Composition

In order to enable the kind of inferencing we have described, composition must have predictable semantics, which we accomplish by defining a restricted composition language of relations and properties. These relations and properties have their own axioms defining what inferences will be drawn from the composition of components.

Relations

We have defined a small set of relations to connect Entities and Events. Keeping the set small — we currently have about eighty — will allow us to maintain detailed axioms for each relation that capture the semantics of the composition of the related components. Writing such axioms for an open-ended set of relations might not be as feasible. The small number of relations also makes it easier for our inexperienced users to learn to use them. Our relations that link an event to an entity describe the participants involved in the event. Our original set was inspired by a comprehensive study of *case roles* in Linguistics [3]. The set has been refined to account for the kinds of relationships expected for our particular event components. Event-to-Entity relations include agent, donor, instrument, object, recipient, result, etc.

To account for relationships between entities, we drew on previous research into the semantics of English noun phrases [2]. Since nouns can represent many things (not just entities) the semantic relationships within noun phrases are a superset of what is required to account for relations between our entities. The set of entity-to-entity relations currently includes content, has-part, location, material, possesses, region, etc.

The choice of relationships between events followed from studies in discourse analysis [1] and process planning [22]. These relations include causes, defeats, enables, entails, inhibits, by-means-of, prevents, resulting-state and subevent.

In addition to the relations among events and entities, we have a very small number of relations that involve roles [13]: relations that link an Entity to the Role it plays (or is intended to play) and between the Role and the Event.

Properties

We also have a small number of *properties*. Properties link entities to values. For example, the *size* of an entity is a property that takes a value. The value can be a cardinal (*25 kilograms*), a scalar (*big relative to housecats*) or a categorical (*brown*).

To define our set of properties, we turned once again to linguistic studies. Whereas events and entities usually surface as verbs and nouns in language, properties are closely related to adjectives. Since adjectives can also represent entities, we restricted our study to those adjectives that ascribe values to features of the nouns they modify. These adjectives are often called *ascriptive* adjectives.

We consulted work in Linguistics on adjective semantics, most notably Dixon [12] and Frawley [15]. We then conducted two exercises to build a list of properties.

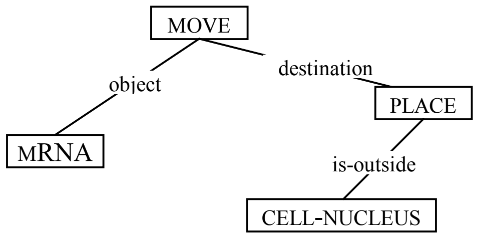
For the first exercise we once again used WordNet, which explicitly distinguishes ascriptive adjectives from non-ascriptive adjectives (called *pertainyms* in WordNet). For the ascriptive adjectives, there are occasionally links to the noun that best describes the "attribute" to which the adjective ascribes a value. For example, the attribute for *large* is *size*. WordNet identifies about 160 unique nouns that are used as attributes. We used these attributes to populate the adjective classes proposed by Dixon, resulting in a first draft of a list of properties.

For the second exercise we once again consulted Roget. As described earlier, although the Roget headwords are nouns, some of them more naturally describe events and have rich verb paragraphs. For other headwords, the adjective paragraphs are the richest (for example, headword #192: Size). Some of the headwords so naturally indicate properties that the verb paragraphs contain little more than "be <adjective>". In headword #201: Shortness, the first entry in the verb paragraph is "be short". Our experiment pulled headwords whose verb paragraphs begin with copular adjectival complementation phrases[2], producing a list of candidates for properties. This test also singles out ascriptive adjectives, since nonascriptive adjectives do not appear as copular complements. Unfortunately, the test did not filter out certain relations (such as "be identical to", "be different from", etc.). In the exercise we removed these relations by hand. The result was a list of approximately 230 headwords representing properties.

We then grouped all of the candidate properties into approximately 25 general categories. This final list of properties includes such properties as age, area, capacity, color, length, shape, size, smell and wetness.

A Simple Example of Composition

Consider the simple example of messenger RNA (mRNA) leaving a cell nucleus. In our interface, the user might describe this action by making an MRNA the object of a MOVE whose destination is outside a CELL-NUCLEUS:



The composition is richer than the mere connection of components due to the extra inferences KM can draw from the connection. Since the destination of the MOVE is outside some place, KM will recognize that this MOVE satisfies the definition of EXIT and will reclassify this instance of MOVE to be an instance of EXIT. Through the semantics of EXIT, KM will infer that prior to the EXIT the MRNA was inside the CELL-NUCLEUS, and that the MRNA must have EXITed through a portal in the CELL-NUCLEUS. KM can also infer that CELL-NUCLEUS must be playing the role of CONTAINER, and that its content prior to the EXIT included the MRNA. In simulating the EXIT, KM will assert that the location of the MRNA is a PLACE outside the CELL-NUCLEUS in the situation immediately following the EXIT.

USING THE LIBRARY

User Interface

Access to the component library is through a web-based tool for building compositions through graph operations:

- add a component to a graph
- connect two components with a relation
- specialize a component to one of its subclasses
- unify two component instances

The use of this tool for knowledge entry is described in detail in [8].

Searching or Browsing the Library

The main disadvantage in restricting ourselves to a small number of generic components is that the library will probably not have a component that exactly matches the concept a user is looking for. The library interface, then, must make it easy for the user to find a close enough match. Our interface supports two modes of access to the library: a tree-based browser and a search tool.

Browsing the Library

Since the components are arranged hierarchically in the library, they can be browsed in the form of a tree. Our library browser allows the user to selectively expand the tree to view a component's subclasses. Since the library is small (by design) and we have attempted to make the component names intuitive and transparent, browsing the library through the tree is feasible. Nonetheless, given that our users are expected to have little or no experience with concept hierarchies, we believe it will be easier for them to find components through a search facility.

Searching the Library

The SHAKEN interface allows two kinds of searching: token match searching and semantic matching.

Token matching will return a component whose name exactly matches (or contains) the search term.

Semantic matching traverses the WordNet hierarchy for terms semantically related to components.

As part of the documentation for each component we have identified the WordNet entries that most closely match the semantics of the component. Our WordNet-based search tool finds the search term in WordNet, then climbs the hierarchy of hypernyms (more general terms) finding all components listing those hypernyms in their documentation. Table 2 shows examples of search terms and their results.

search term	*components found*
assemble	ATTACH, CREATE, COME-TOGETHER, MOVE-TOGETHER
mend	REPAIR
gum-up	OBSTRUCT, BLOCK
busted	BE-BROKEN, BE-RUINED

Table 2: Examples of search terms and components found

One of the advantages of semantic searching is that results are sorted on the WordNet distance between the search term and the component, and on the depth of the component in our hierarchy. This gives preference to more specific library

[2] a phrase of the form "<copula> <adjective>", where the copulas include "be", "become", "seem", etc.

components, meaning the user is more likely to choose a more specific (and therefore more semantically loaded) component than if she browsed top-down through the tree.

Documentation

One of the disadvantages of giving components names that are also English words is that users may have different biases about the senses of those words. These expectations may clash with the semantics of the components. This problem underlines the need for good documentation. Our documentation for components includes several things:

- Definitions. Given our particular users, it is important to have simple, non-technical definitions that describe all of (and only) the meaning of each component.
- One-line glosses. Early experiments have shown that users often accept or reject a component based solely on its name in a list. In our web-based interface one-line glosses are displayed as the mouse hovers over a component name.
- More detailed documentation. We also document the full semantics of components, including participants in an event, subevents, parts of an entity, etc.
- Examples. Several examples of varying complexity help show the intended use of components.
- Neighboring concepts. We are adding information to the documentation on "neighboring concepts" (similar components and how they differ from each other).

All documentation is available through the user interface, which can also show a graph representation of components in the library. In the graph the user can choose to see all, none or any subset of the links between the component and components connected to it through our relations.

EVALUATION

We have conducted three experiments in which biologists with no experience in knowledge engineering were asked to encode knowledge using our library and *SHAKEN*. In the first two experiments, users were given roughly one day of training on how to use the system and one day to encode a biological process (DNA transcription). In the third experiment we provided roughly one week of training. Users then (over eight weeks) encoded the knowledge from one chapter of a textbook in Cell Biology. All interaction between the users and developers was indirect, mediated by an impartial "gatekeeper" knowledge engineer.

In all three experiments the users have shown that our library search facility is able to guide them to generic components that they are willing to accept as abstractions of concepts they wish to encode. For example, our users were comfortable defining biological processes in terms of such generic actions as COLLIDE, SLIDE, ATTACH, RELEASE, etc.

At the end of each experiment, users were also asked to fill out a questionnaire. On average users found it moderately easy (slightly over 3 on a scale of 1-to-5) to find relevant components in the library. They found it easy (4 on a scale of 1-to-5) to understand the components. They found the components useful (4) for representing knowledge. They found the restricted language of relations easy (4) to understand and use. They found it moderately difficult (slightly under 3) to cast biological knowledge in terms of the components and relations in the library.

More objective data on the quality of the representations of the knowledge built by the users is being collected.

LIMITATIONS AND FUTURE WORK

It is part of our claim that arbitrarily complex knowledge can be represented through the composition of simple generic components. The goal of knowledge reuse, however, suggests that the library will be even more powerful if it allows users to compose these more complex compositions themselves. One of our main tasks ahead is to populate the library with more complex (yet still general) components, such as processes of DELIVERY, PRODUCTION, COMMUNICATION, etc. We are also investigating ways to automate the composition of more complex components, such as through the use of interface templates.

Users identified several ways in which the library could be improved. For example, biologists are often interested in representing *functional* aspects of processes, not just *physical/spatial* descriptions. Including role concepts and the notion of *purpose* is currently making the encoding of functional knowledge easier. We have made extensions to our relationship language and are continuing to expand our role hierarchy and our generic entity hierarchy to admit functional representations more easily.

Experiments also underlined the importance of simple component names that do not have strong connotations in a particular domain. Our general actions of REPLICATE and TRANSCRIBE caused confusion among the biologists. We are currently reviewing the names (and documentation) of all our components. We are also reviewing components that users found unintuitive for other reasons. For example, users were not interested in the distinction between MOVE and its subclasses MOVE-FROM and MOVE-TO. We will likely remove these components for the sake of simplicity.

SUMMARY

In this paper we have described the process of building a library of knowledge components under the pressures imposed by our intended audience: domain experts with no experience in ontologies or knowledge engineering. In order to make the library accessible, we have taken inspiration from linguistic resources and built hooks to language into the components. In order to achieve power through composition of components, we have limited the library to a small number of components and relations with rich semantics. Preliminary trials with users inexperienced in knowledge engineering have been promising, giving us hope that domain experts will soon be able to encode their expertise in powerful knowledge bases.

ACKNOWLEDGMENTS

The authors are indebted to Art Souther, James Fan, Paul Navratil, Dan Tecuci, Peter Yeh, Marwan Elrakabawy, Sarah Tierney, John Thompson, Vinay Chaudhri, Andres Rodriguez, Jérôme Thoméré, Yolanda Gil, Jim Blythe, Jihie Kim, Pat Hayes and Paul Cohen for input on the construction and use of the component library.

Support for this research is provided by a contract from Stanford Research Institute as part of DARPA's Rapid Knowledge Formation project. This material is based upon work supported by the Space and Naval Warfare Systems Center - San Diego under Contract No. N66001-00-C-8018.

REFERENCES

1. Barker, K., and Szpakowicz, S. Interactive Semantic Analysis of Clause-Level Relationships, in *Proceedings of PACLING '95* (Brisbane, April 1995).
2. Barker, K., and Szpakowicz, S. Semi-Automatic Recognition of Noun Modifier Relationships, in *Proceedings of COLING-ACL '98* (Montréal, August 1998), 96-102.
3. Barker, K., Copeck, T., Delisle, S. & Szpakowicz, S. Systematic Construction of a Versatile Case System. *Journal of Natural Language Engineering 3, 4* (December 1997), 279-315.
4. BNC. The British National Corpus. Available at http://info.ox.ac.uk/bnc/, 2001.
5. Clark, P., and Porter, B. Building Concept Representations from Reusable Components, in *Proceedings of AAAI '97* (Providence RI, July 1997), AAAI Press, 369-376.
6. Clark, P. and Porter, B. KM – The Knowledge Machine: User's Manual and Situations Manual. Available at http://www.cs.utexas.edu/users/mfkb/RKF/km.html, 2001.
7. Clark, P., Thompson, J. and Porter, B. Knowledge Patterns, in *Proceedings of KR-2000* (Breckenridge CO, April 2000), Morgan Kaufmann, 591-600.
8. Clark, P., Thompson, J., Barker, K., Porter, B., Chaudhri, V., Rodriguez, A., Thoméré, J., Gil, Y. and Hayes, P. Knowledge Entry as the Graphical Assembly of Components: The SHAKEN System, in *Proceedings of K-CAP 2001* (Victoria BC, October 2001).
9. COLLINS. *The Collins English Dictionary.* William Collins Sons, 1979. At the Linguistic Data Consortium, http://www.ldc.upenn.edu/Catalog/LDC93T1.html, 1993.
10. CYC. The Cyc Upper Ontology. Available at http://www.cyc.com/cyc-2-1/index.html, 2001.
11. DARPA. The Rapid Knowledge Formation Project. Available at http://reliant.teknowledge.com/RKF/, 2000.
12. Dixon, R. M. W. *Where Have all the Adjectives Gone?* Mouton, The Hague, 1982.
13. Fan, J., Barker, K., Porter B. and Clark, P. Representing Role Concepts, in *Proceedings of K-CAP 2001* (Victoria BC, October 2001).
14. Fikes, R. and Farquhar, A. Distributed Repositories of Highly Expressive Reusable Ontologies. *IEEE Intelligent Systems 14, 2* (March/April 1999), 73-79.
15. Frawley, W. *Linguistic Semantics*. Lawrence Erlbaum Associates, Publishers, Hillsdale NJ, 1992.
16. Knight, K. and Luk, S. Building a Large-Scale Knowledge Base for Machine Translation, in *Proceedings of AAAI '94* (Seattle WA, July-August 1994), AAAI Press, 773-778.
17. Lester, J. and Porter, B. Developing and Empirically Evaluating Robust Explanation Generators: The KNIGHT Experiments. *Computational Linguistics 23, 1* (1997), 65-101.
18. Lenat, D. Cyc: A Large-Scale Investment in Knowledge Infrastructure. *Communications of the ACM 38, 11* (November 1995), 33-38.
19. Lenat, D. and Guha, R. *Building Large Knowledge Based Systems.* Addison Wesley, Reading, Massachusetts, 1990.
20. Lloyd, S. M. *Roget's Thesaurus*. Longman, Essex, 1982.
21. Miller, G. A. (ed.). WordNet: An Online Lexical Database. *International Journal of Lexicography 3, 4* (Winter 1990).
22. Narayanan, S. Reasoning about Actions in Narrative Understanding, in *Proceedings of IJCAI'99* (Stockholm, August 1999), Morgan Kaufmann, 350-358.
23. Parmar, A. The Representation of Actions in KM and Cyc. Department of Computer Science, Stanford University technical report (forthcoming), 2001.
24. Rickel, J. and Porter, B. Automated Modeling of Complex Systems to Answer Prediction Questions. *Artificial Intelligence Journal 93, 1-2* (1997), 201-260.
25. Schank, R. C. *Conceptual Information Processing*. North-Holland Publishing Company, Amsterdam, 1975.
26. Schank, R. C. and Abelson, R. P. *Scripts, Plans, Goals and Understanding*. Erlbaum, Hillsdale NJ, 1977.
27. SRI. SRI's Rapid Knowledge Formation Team. Available at http://www.ai.sri.com/~rkf, 2001.
28. SRI. Proposal to DARPA's Rapid Knowledge Formation Project. Available at http://reliant.teknowledge.com/RKF/proposals/SRI/SRIproposal.htm, 2000.
29. Summers, D. (ed.). Longman Dictionary of Contemporary English: New Edition. Longman, Essex, 1987.