Enabling Technology for Knowledge Sharing

Robert Neches, Richard Fikes, Tim Finin, Thomas Gruber, Ramesh Patil, Ted Senator, and William R. Swartout

Building new knowledge-based systems today usually entails constructing new knowledge bases from scratch. It could instead be done by assembling reusable components. System developers would then only need to worry about creating the specialized knowledge and reasoners new to the specific task of their system. This new system would interoperate with existing systems, using them to perform some of its reasoning. In this way, declarative knowledge, problem-solving techniques, and reasoning services could all be shared among systems. This approach would facilitate building bigger and better systems cheaply. The infrastructure to support such sharing and reuse would lead to greater ubiquity of these systems, potentially transforming the knowledge industry. This article presents a vision of the future in which knowledge-based system development and operation is facilitated by infrastructure and technology for knowledge sharing. It describes an initiative currently under way to develop these ideas and suggests steps that must be taken in the future to try to realize this vision.

Ever since the mid-seventies, researchers have recognized that capturing knowledge is the key to building large and powerful AI systems. In the years since, we have also found that representing knowledge is difficult and time consuming. Although we have developed tools to help with knowledge acquisition, knowledge base construction remains one of the major costs in building an AI system: For almost every system we build, a new knowledge base must be constructed from scratch. As a result, most systems remain small to medium in size. Even if we build several systems within a general area, such as medicine or electronics diagnosis, significant portions of the domain must be rerepresented for every system we create.

The cost of this duplication of effort has been high and will become prohibitive as we attempt to build larger and larger systems. To overcome this barrier and advance the state of the art, we must find ways of preserving existing knowledge bases and of sharing, reusing, and building on them.

This article describes both near- and long-term issues underlying an initiative to address these concerns. First, we discuss four bottlenecks to sharing and reuse, present a vision of a future in which these bottlenecks have been ameliorated, and touch on the efforts of the initiative's four working groups to address these bottlenecks. We then elaborate on the vision by describing the model it implies for how knowledge bases and knowledge-based systems could be structured and developed. This model involves both infrastructure and supporting technology. The supporting technology is the topic of our near-term interest because it is critical to enabling the infrastructure. Therefore, we return to discussing the efforts of the four working groups of our initiative, focusing on the enabling technology that they are working to define. Finally, we consider topics of longer-range interest by reviewing some of the research issues raised by our vision.

Sharing and Reuse

There are many senses in which the work that went into creating a knowledge-based system can be shared and reused. Rather than mandating one particular sense, the model described in this article seeks to support several of them. One mode of reuse is through the exchange of techniques. That is, the content of some module from the library is not directly used, but the approach behind it is communicated in a manner that facilitates its reimplementation. Another mode of reuse is through the inclusion of source specifications. That is, the content of some module is copied into another at design time and compiled (possibly after extension or revision) into the new component. A third mode is through the run-time invocation of external modules or services. That is, one module invokes another either as a procedure from a function library or through the maintenance of some kind of client-server relationship between the two (Finin and Fritzson 1989).

These modes of reuse do not work particularly smoothly today. Explaining how to reproduce a technique often requires communicating subtle issues that are more easily expressed formally; whether stated formally or in natural language, the explanations require shared understanding of the intended interpretations of terms. The reuse of source specifications is only feasible to the extent that their model of the world is compatible with the intended new use. The reuse of external modules is feasible only to the extent that we understand what requests the modules are prepared to accept. Let us consider these complexities in more detail by reviewing four critical impediments to sharing and reuse.

Impediment 1. Heterogeneous Representations: There are a wide variety of approaches to knowledge representation, and knowledge that is expressed in one formalism cannot

Attempting to move beyond the capabilities of current knowledge-based systems mandates knowledge bases that are substantially larger than those we have today.

0738-4602/91/$4.00 ©1991 AAAI

Instead, the process of building a knowledge-based system will start by assembling reusable components.

directly be incorporated into another formalism. However, this diversity is inevitable—the choice of one form of knowledge representation over another can have a big impact on a system's performance. There is no single knowledge representation that is best for all problems, nor is there likely to be one. Thus, in many cases, sharing and reusing knowledge will involve translating from one representation to another. Currently, the only way to do this translating is by manually recoding knowledge from one representation to another. We need tools that can help automate the translation process.

Impediment 2. Dialects within Language Families: Even within a single family of knowledge representation formalisms (for example, the KL-One family), it can be difficult to share knowledge across systems if the knowledge has been encoded in different dialects. Some of the differences between dialects are substantive, but many involve arbitrary and inconsequential differences in syntax and semantics. All such differences, substantive or trivial, impede sharing. It is important to eliminate unnecessary differences at this level.

Impediment 3. Lack of Communication Conventions: Knowledge sharing does not necessarily require a merger of knowledge bases. If separate systems can communicate with one another, they can benefit from each other's knowledge without sharing a common knowledge base. Unfortunately, this approach is not generally feasible for today's systems because we lack an agreed-on protocol specifying how systems are to query each other and in what form answers are to be delivered. Similarly, we lack standard protocols that would provide interoperability between knowledge representation systems and other, conventional software, such as database management systems.

Impediment 4. Model Mismatches at the Knowledge Level: Finally, even if the language-level problems previously described are resolved, it can still be difficult to combine two knowledge bases or establish effective communications between them. These remaining barriers arise when different primitive terms are used to organize them; that is, if they lack shared vocabulary and domain terminology. For example, the type hierarchy of one knowledge base might split the concept Object into Physical-Object and Abstract-Object, but another might decompose Object into Decomposable-Object, Nondecomposable-Object, Conscious-Being, and Non-Conscious-Thing. The absence of knowledge about the relationship between the two sets of terms makes it difficult to reconcile them. Sometimes these differences reflect differences in the intended purposes of the knowledge bases. At other times, these differences are just arbitrary (for example, different knowledge bases use Isa, Isa-kind-of, Subsumes, AKO, or Parent relations, although their real intent is the same). If we could develop shared sets of explicitly defined terminology, sometimes called *ontologies*, we could begin to remove some of the arbitrary differences at the knowledge level. Furthermore, shared ontologies could provide a basis for packaging knowledge modules—describing the contents or services that are offered and their ontological commitments in a composable, reusable form.

A Vision: Knowledge Sharing

In this article, we present a vision of the future in which the idea of knowledge sharing is commonplace. If this vision is realized, building a new system will rarely involve constructing a new knowledge base from scratch. Instead, the process of building a knowledge-based system will start by assembling reusable components. Portions of existing knowledge bases would be reused in constructing the new system, and special-purpose reasoners embodying problem-solving methods would similarly be brought in. Some effort would go into connecting these pieces, creating a "custom shell" with preloaded knowledge. However, the majority of the system development effort could become focused on creating only the specialized knowledge and reasoners that are new to the specific task of the system under construction. In our vision, the new system could interoperate with existing systems and pose queries to them to perform some of its reasoning.

Furthermore, extensions to existing knowledge bases could be added to shared repositories, thereby expanding and enriching them.

Over time, large rich knowledge bases, analogous to today's databases, will evolve. In this way, declarative knowledge, problem-solving techniques and reasoning services could all be shared among systems. The cost to produce a system would decrease. To the extent that well-tested parts were reused, a system's robustness would increase.

For end users, this vision will change the face of information systems in three ways. First, it will provide sources of information that serve the same functions as books and libraries but are more flexible, easier to update, and easier to query. Second, it will enable the construction and marketing of prepackaged knowledge services, allowing users to invoke (rent or buy) services. Third, it will make it possible for end users to tailor large systems to their needs by assembling knowledge bases and services rather than programming them from scratch.

We also expect changes and enhancements in the ways that developers view and manipulate knowledge-based systems. In particular, we envision three mechanisms that would increase their productivity by promoting the sharing and reuse of accumulated knowledge. First among these are libraries of multiple layers of reusable knowledge bases that could either be incorporated into software or remotely consulted at execution time. At a level generic to a class of applications, layers in such knowledge bases capture conceptualizations, tasks, and problem- solving methods. Second, system construction will be facilitated by the availability of common knowledge representation systems and a means for translation between them. Finally, this new reuse-oriented approach will offer tools and methodologies that allow developers to find and use library entries useful to their needs as well as preexisting services built on these libraries. These tools will be complemented by tools that allow developers to offer their work for inclusion in the libraries.

The Knowledge-Sharing Effort

We are not yet technically ready to realize this vision. Instead, we must work toward it incrementally. For example, there is no consensus today on the appropriate form or content of the shared ontologies that we envision. For this consensus to emerge, we need to engage in exercises in building shared knowledge bases, extract generalizations from the set of systems that emerge, and capture these generalizations in a standard format that can be interpreted by all involved. This process requires the development of some agreed-on formalisms and conventions at the level of an interchange format between languages or a common knowledge representation language.

Simply enabling the ability to share knowledge is not enough for the technology to have full impact, however. The development and use of shared ontologies cannot become cost effective unless the systems using them are highly interoperable with both AI and conventional software, so that large numbers of systems can be built. Thus, software interfaces to knowledge representation systems are a crucial issue.

The Knowledge-Sharing Effort, sponsored by the Air Force Office of Scientific Research, the Defense Advanced Research Projects Agency, the Corporation for National Research Initiatives, and the National Science Foundation (NSF), is an initiative to develop the technical infrastructure to support the sharing of knowledge among systems. The effort is organized into four working groups, each of which is addressing one of the four impediments to sharing that we outlined earlier. The working groups are briefly described here and in greater detail later in the article.

The Interlingua Working Group is developing an approach to translating between knowledge representation languages. Its approach involves developing an intermediary language, a *knowledge interchange format* or *interlingua*, along with a set of translators to map into and out of it from existing knowledge representation languages. To map a knowledge base from one representation language into another, a system builder would use one translator to map the knowledge base into the interchange format and another to map from the interchange format back out to the second language.

The Knowledge Representation System Specification (KRSS) Working Group is taking another, complementary tack toward promoting knowledge sharing. Rather than translating between knowledge representation languages, the KRSS group is seeking to promote sharing by removing arbitrary differences among knowledge representation languages within the same paradigm. This group is currently working on a specification for a knowledge representation system that brings together the best features of languages developed within the KL-One paradigm. Similar efforts for other families of languages are expected to follow.

The External Interfaces Working Group is

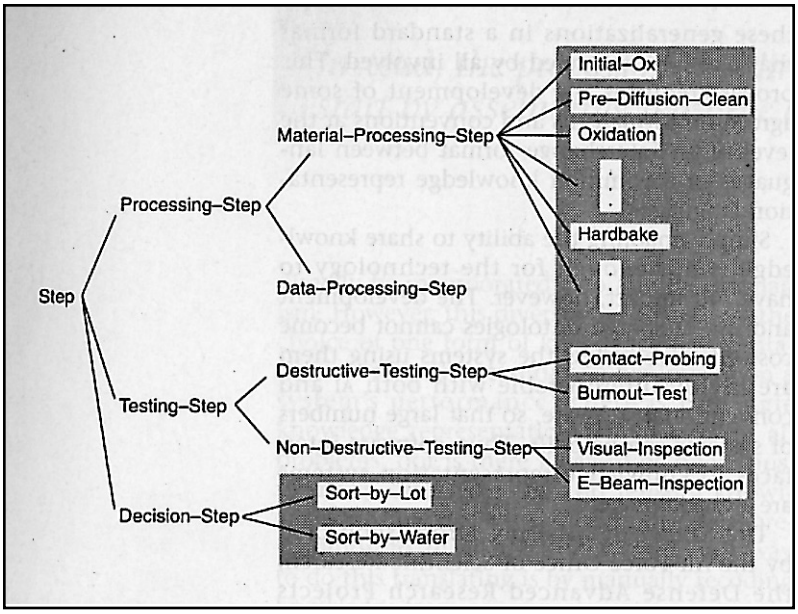


Figure 1. The MKS Ontology of Manufacturing Operations, Elaborated with Knowledge Specific to Semiconductor Manufacturing.

Ontologies such as this one, in effect, lay the ground rules for modeling a domain by defining the basic terms and relations that make up the vocabulary of this topic area. These ground rules serve to guide system builders in fleshing out knowledge bases, building services that operate on knowledge bases, and combining knowledge bases and services to create larger systems. For one system to make use of either the knowledge or reasoners of another system, the two must have consistent ontologies.

investigating yet another facet of knowledge sharing. It is developing a set of protocols for interaction that would allow a knowledge-based system to obtain knowledge from another knowledge-based system (or, possibly, a conventional database) by posting a query to this system and receiving a response. The concerns of this group are to develop the protocols and conventions through which such an interaction could take place.

Finally, the Shared, Reusable Knowledge Bases Working Group is working on overcoming the barriers to sharing that arise from lack of consensus across knowledge bases on vocabulary and semantic interpretations in domain models. As mentioned earlier, the ontology of a system consists of its vocabulary and a set of constraints on the way terms can be combined to model a domain. All knowledge systems are based on an ontology, whether implicit or explicit. A larger knowledge system can be composed from two smaller ones only if their ontologies are consistent. This group is trying to ameliorate problems of inconsistency by fostering the evolution of common, shareable ontologies. A number of candidate reusable ontologies are expected to come from this work. However, the ultimate contribution of the group lies in developing an understanding of the group processes that evolve such products and the tools and infrastructure needed to facilitate the creation, dissemination, and reuse of domain-oriented ontologies.

Architectures of the Future

In this section, we elaborate on our vision by describing what we hope to enable concerning both knowledge bases and the systems that use them. In doing so, we look at how they are structured and the process by which they will be built. We also consider the relationship of this vision to other views that have been offered, such as Guha and Lenat's (1990) Cyc effort, Stefik's (1986) notion of Knowledge Media, and Kahn's notion of Knowbots (Kahn and Cerf 1988). Finally, we offer a view of the range of system models that this approach supports.

Structural and Development Models for Knowledge Bases

In a AAAI-90 panel on software engineering, John McDermott (1990) described how AI could make software development easier: Write programs to act as frameworks for handling instances of problem classes.

Knowledge-based systems can provide such frameworks in the form of top-level declarative abstraction hierarchies, which an application builder elaborates to create a specific system. Essentially, hierarchies built for this purpose represent a commitment to provide specific services to applications that are willing to adopt their model of the world.

When these top-level abstraction hierarchies are represented with enough information to lay down the ground rules for modeling a domain, we call them ontologies. An ontology defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary. An example is the MKS generic model of manufacturing steps (Pan, Tenenbaum, and Glicksman 1989), illustrated in figure 1 along with a set of application-specific extensions for semiconductor manufacturing. The frame hierarchy in MKS defines classes of concepts that the system's reasoning modules (for example, schedulers and diagnosers) are prepared to operate on. The slots and slot restrictions on these frames define how one must model a particular manufacturing domain to enable the use of these modules.

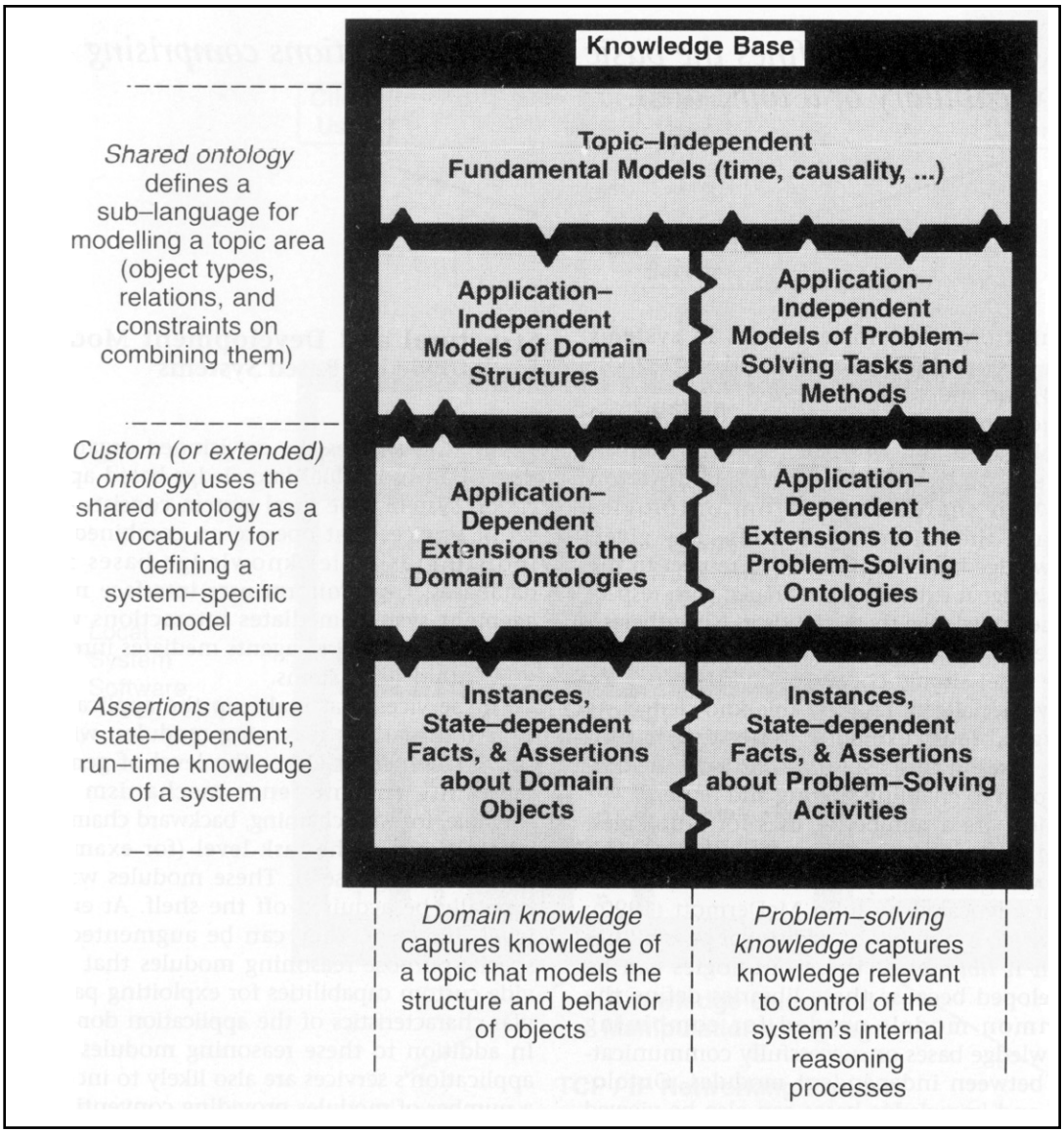


Figure 2. The Anatomy of a Knowledge Base.

Application systems contain many different kinds and levels of knowledge. At the top level are ontologies, although often represented only implicitly in many of today's systems. The top-level ontologies embody representational choices ranging from topic independent (for example, models of time or causality) to topic specific but still application-independent knowledge (for example, domain knowledge about different kinds of testing operations represented in a manufacturing system or problem-solving knowledge about hypothesis classes in a diagnostic system). This top level of knowledge is elaborated by more application-specific models (for example, knowledge about chip-testing operations in a specific manufacturing application or failure modes in a circuit diagnosis system). Together, they define how a particular application describes the world. At the bottom level, assertions using the vocabulary of these models capture the current state of the system's knowledge. Knowledge at the higher levels, being less specialized, is easier to share and reuse. Knowledge at the lower levels can only be shared if the other system accepts the models in the levels above.

The MKS example is hardly unique. A number of systems have been built in a manner consistent with this philosophy, for example, FIRST-CUT and NEXT-CUT (Cutkosky and Tenenbaum 1990), QPE (Forbus 1990), Cyc (Guha and Lenat 1990), ARIES (Johnson and Harris 1990), SAGE (Roth and Mattis 1990), Carnegie Mellon University's factory scheduling and project management system (Sathi, Fox, and Greenberg 1990), KIDS (Smith 1990), and EES (Swartout and Smoliar 1989). The notion that generic structure can be exploited in building specialized systems has been argued for a long time by Chandrasekaran (1983, 1986) and more recently by Steels (1990). The notion has also long been exploited in knowledge-

An ontology defines the basic terms and relations comprising the vocabulary of a topic area…

acquisition work, for example, in systems such as MORE (Kahn, Nowlan, and McDermott 1984) and ROGET (Bennett 1984).

The range of knowledge captured with ontologies is described in figure 2. There is some fuzziness in the figure's distinction between shared and custom ontologies because they are relative terms—any given knowledge base is custom with respect to the knowledge it extends and is shared with respect to the knowledge that extends it. Nevertheless, the essential idea is that application knowledge bases should consist of layers of increasingly specialized, less reusable knowledge. As is argued later, explicitly organizing knowledge bases in this fashion is a step on the critical path to enabling sharing and reuse.

There are a number of uses for ontologies. The primary uses today are in defining knowledge-based system frameworks in the spirit advocated by John McDermott (1990). However, a number of additional possibilities open if *libraries* of these ontologies can be developed because these libraries define the common models needed for combining knowledge bases or successfully communicating between independent modules. Ontologies and knowledge bases can also be viewed as ends in themselves, that is, as repositories of information in the spirit suggested by Stefik's (1986) discussion of knowledge media.

Although every knowledge-based system implicitly or explicitly embodies an ontology, ontologies are rarely shared across systems. Commonalities among existing systems can be identified and made shareable. For example, Stanford's Summer Ontology Project found that several of the collaborators had built systems that used models of mechanical devices based on concepts such as module, port, and connection (Gruber 1991). However, each system used slightly different names and formalisms. An ontology for lumped element models that defines these concepts with consistent, shareable terminology is under construction. A library of such shared ontologies would facilitate building systems by reducing the effort invested in reconciliation and reinvention.

Structural and Development Models for Knowledge-Based Systems

Figure 3 illustrates the envisioned organization of an individual knowledge-based application system. The local system consists of a set of services that operate on combined (or indistinguishable) knowledge bases and databases. One uniform user interface management system mediates interactions with humans, and a set of agents mediates interaction with other systems.

The services that make up the application consist of various reasoning modules, which can be defined at either the level of generic reasoning and inference mechanism (for example, forward chaining, backward chaining, unification) or the task level (for example, planners, diagnosers). These modules would typically be acquired off the shelf. At either level, however, they can be augmented by special-purpose reasoning modules that provide custom capabilities for exploiting particular characteristics of the application domain. In addition to these reasoning modules, the application's services are also likely to include a number of modules providing conventional capabilities, such as spreadsheets, electronic mail, hypermedia, calendar systems, statistical packages, and accounting systems.

To perform the tasks of the overall system, modules will need to interact internally (that is, knowledge base to knowledge base) and externally (that is, knowledge base–database to other knowledge-based systems or arbitrary outside software applications). For external interaction, the modules will need a language for encoding their communications. SQL serves this function for conventional database interactions, and it appears likely that it will continue to be used in the future. We call the analogous programmatic interface for knowledge-based applications KQML (knowledge query and manipulation language). KQML will consist of a language for specifying wrappers that define messages communicated between modules. These wrappers will surround declarations that will express whatever knowledge

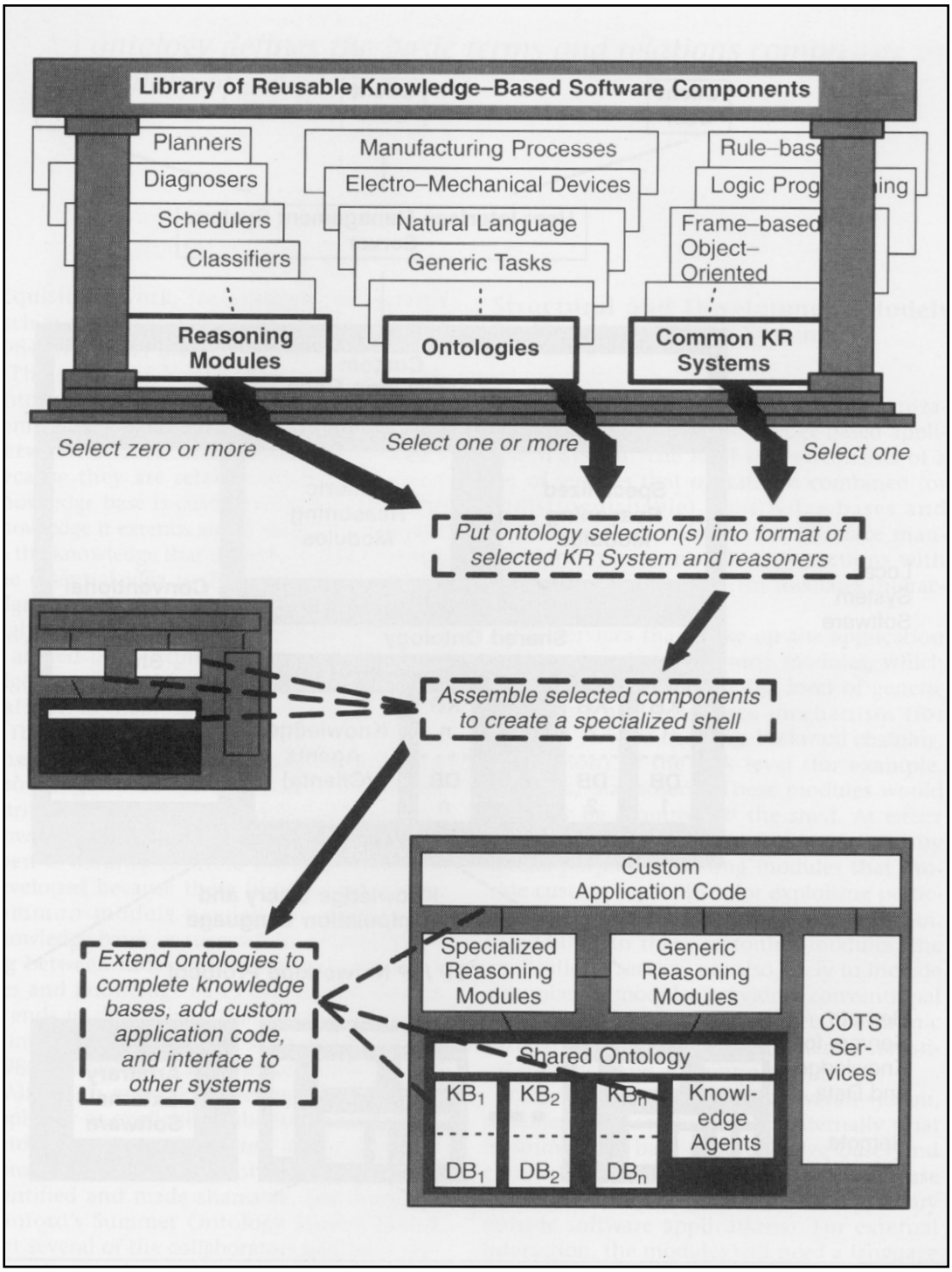


Figure 3. Architecture of a Knowledge-Based System.

We envision that knowledge-based systems will be assembled from components rather than built from scratch. The components include a framework for local system software in which one or more local knowledge bases are tied to a shared ontology. Remote knowledge bases can be accessed and are understood by the local system by virtue of being tied in to the ontology. Specialized reasoning modules (for example, a diagnostic system) and generic reasoning systems (that is, a representation system) are also tied to the knowledge base(s) through the ontology. In turn, these systems are glued together with conventional services through specialized, custom application code. Larger systems can be obtained from smaller ones in this architecture by either expanding the contents of a local system or interlinking multiple systems built in this fashion.

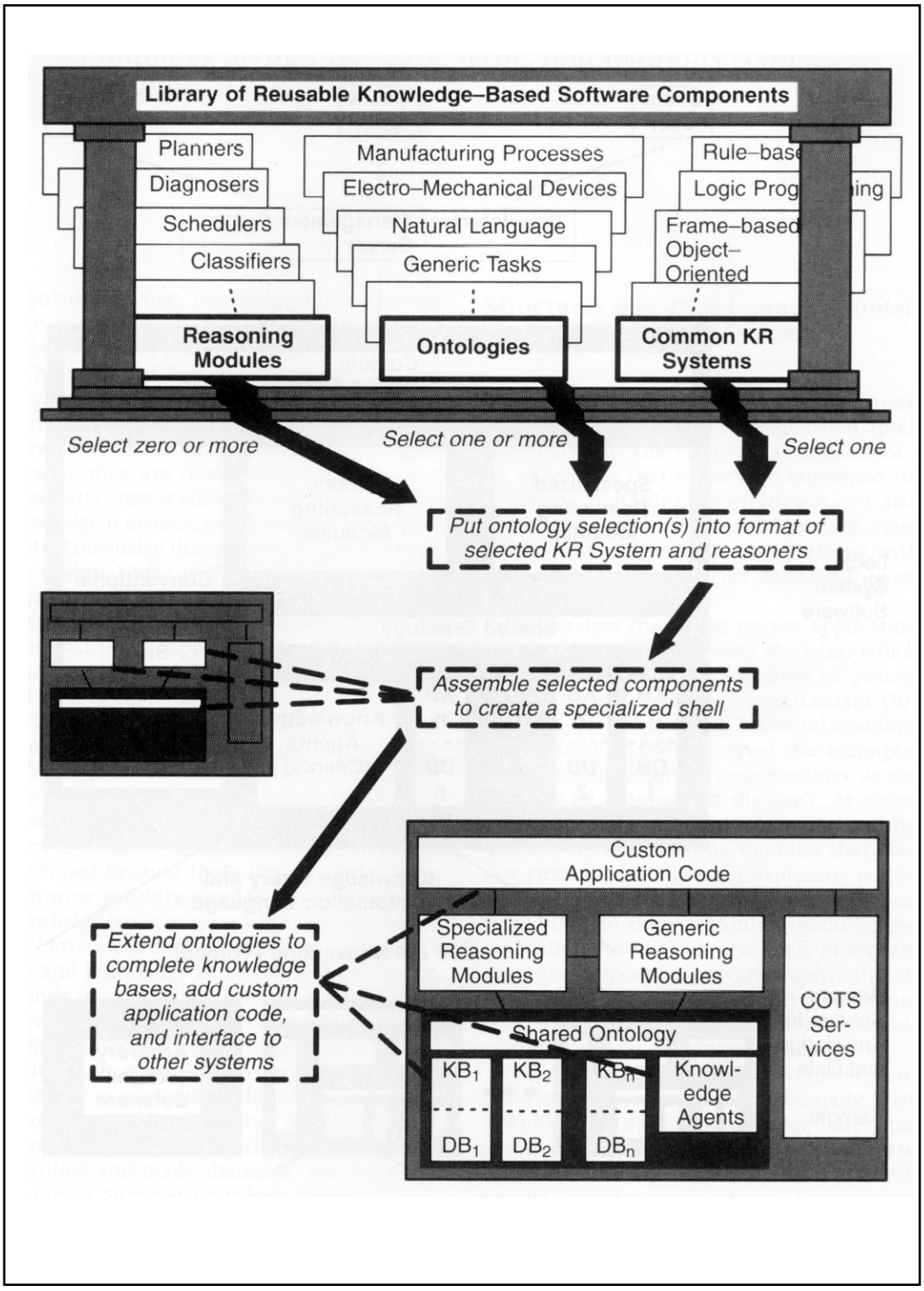


Figure 4. Envisioned Phases in Defining a Knowledge-Based System.

With libraries of reusable knowledge-based software components, building an application could become much more of a configuration task and, correspondingly, less of a programming activity. System builders could select specialized reasoning modules and ontologies, converting them to the format required by the selected knowledge representation if they were not already in this format. This approach gives them a specialized shell with some knowledge built in. This custom shell can then be utilized to build the application from a higher-level starting point.

the sending module must pass to the receiving module.

The uniform KQML communication protocol will facilitate modular construction of system components and, thus, facilitate the development of libraries of reusable components. Such libraries will contain generic reasoners such as truth maintenance systems, task-specific reasoners (planners, diagnosers, and so on), and domain-specific reasoners (for example, electric circuit simulators). The libraries will also contain a number of ontologies covering both structural knowledge and problem-solving knowledge. Some will be domain oriented, and some will correspond to particular reasoners. Also residing in these libraries will be a collection of knowledge representation system implementations to choose from. We expect that several different representational paradigms, or families, might be available in the library. For each family, we expect multiple implementations of a common core language, similar to Common Lisp (Steele 1984), which has a core specification and several competing implementations of this specification, each of which offers various performance, environment, and function enhancements.

As figure 4 illustrates, application-specific systems will be developed by assembling components from a library into a customized shell, which is then used to develop the application. The first task of system developers would be to configure this specialized shell. This process will involve selecting ontologies, specialized reasoning modules, and a knowledge representation system from the library. As in any configuration task, there are likely to be constraints that must be respected. For example, a particular specialized scheduling module might assume that tasks to be scheduled are modeled according to a particular ontology. The entries in the library must provide enough information about themselves to allow system developers to understand these constraints.

Once the ontologies, specialized reasoning modules, and a knowledge representation system have been selected and assembled, the system developers have a specialized shell for their application. This shell differs from today's shells in that it will come with built-in knowledge, not just specialized programming features. The system developers' tasks are then to flesh out the knowledge base, add whatever custom application code is necessary, and write software interfaces to any other systems the new application will work with.

In configuring this specialized shell, it would be highly desirable if there were support for translation between representations.

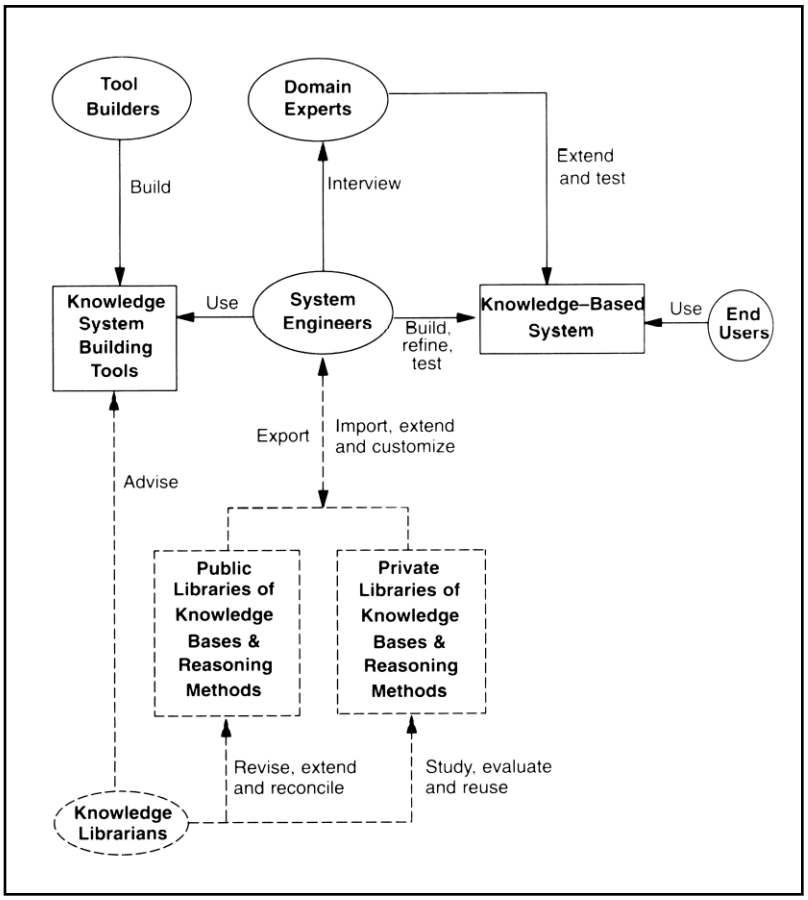


Figure 5. Current versus Envisioned Models of the AI Software Life Cycle.

Adding knowledge libraries represents a significant change in methodology for building knowledge-based systems. Knowledge librarians, a new category of participant in the process, would ensure that submitted ontologies and reasoners were ready for release. They would help system engineers browse the library and select modules. They would also help tool builders create tools and development environments to assist in these activities.

Then, developers would not be obliged to use the original implementation if their application had different performance or function requirements. For example, a common ontology of electromechanical devices could serve DRAMA (Harp et al. 1991), which analyzes logistical implications of design changes, and COMET (Feiner and McKeown 1990), which generates multimedia how-to presentations about maintenance tasks. COMET needs device models to access related device components and primarily requires efficient storage and retrieval from its representation language. In contrast, DRAMA analyzes implications of large amounts of changing knowledge and, therefore, demands efficient inference mechanisms.

Note that the approach we have been describing scales with the evolution of infras-

Adding a knowledge library represents a significant change to knowledge base methodology.

tructure for knowledge sharing. It is feasible with existing technology. If and when common knowledge representation systems become available, their use would broaden the portability and reusability of a given library. Similarly, the development and dissemination of techniques for translation between different representation languages would also broaden the use of a library.

Figure 5 contrasts our model of the life cycle for knowledge-based software with the approach followed in expert system software today. The notion of libraries is a key difference. In today's models of expert system software development, exemplified by Waterman's (1986) book on expert systems, there are at least four classes of participants in the life cycle. Tool builders create shells and development environments. System engineers take these tools and create an initial version of a knowledge-based system by interviewing domain experts. Then, together with the domain experts, the system engineers test, extend, and refine the system. Finally, end users put the system to use. The vision we propose changes the nature of the system engineer's work and has the potential to merge roles by giving domain experts more ability to act as system engineers. It also adds a new participant to the infrastructure: a knowledge librarian, who serves as a keeper of reusable ontologies and implementations of specialized reasoning methods.

The knowledge librarian works with tool builders on tools and development environments to help system engineers browse and select modules from the library. System engineers import, extend, and customize these modules. They can retain the customized elements in private libraries for sharing and reuse within a subcommunity. Alternatively, they can offer their developments to the library for possible inclusion as extensions to the existing set of knowledge.

One of the crucial functions that must be performed in the management of a knowledge library is the determination of when submitted knowledge is ready for release. If knowledge is simply added and edited by all participants without some discipline, then it will be difficult to achieve the level of reliability and stability needed for practical software development.

Adding a knowledge library represents a significant change to knowledge base methodology. It transforms the system-building process away from using power tools for constructing essentially custom, one-of-a-kind systems. Instead, system building becomes a process of selecting and combining a range of components. The system engineer becomes much more like a builder of homes and much less like a sculptor. A home builder has a range of components and materials, from bricks and two-by-fours to prefabricated walls or even rooms. A home builder has a choice of building each new home from small components that give a lot of design flexibility or from larger components that trade off reduced design options for reduced construction effort. Similarly, system engineers would find a range of grain sizes among objects in the knowledge library and would be empowered to make analogous choices.

Comparison with Other Visions

The model of knowledge-based systems that we just described bears significant relationships to other notions that have been offered.

One recent example is MCC Corporation's Cyc Project (Guha and Lenat 1990), which seeks to develop an extremely large knowledge base of commonsense knowledge under which all applications would be built. The Cyc Project provides a language, Cyc-L, for implementing its ontology and developing an application-specific knowledge base. Because its scope is so broad, Cyc represents one extreme in the range of efforts compatible with the model we propose. In our scheme, Cyc's knowledge base could be one large entry in the library of components (or, perhaps, it might be broken into several smaller modules, or *microtheories*). Its implementation language, Cyc-L, would be one of the entries in the library of representation systems. If one chose to build a system entirely within Cyc, our model of the development process and that of the Cyc Project are largely consis-

tent. If one wishes to go outside Cyc, our model is complementary. Potential users might use our knowledge-interchange format to translate other knowledge bases into Cyc-L or, conversely, to translate Cyc's knowledge into some other representation system. Alternatively, they might use external protocols to access a module built in Cyc and build other modules through other means.

If successful, our approach would help make the knowledge in Cyc accessible even to those system developers who for one reason or another do not choose to use the whole Cyc knowledge base or represent their systems in Cyc-L.

However, the model we propose also differs significantly from the Cyc effort. Among other things, it allows the development of large systems without having to first commit to a particular knowledge representation formalism; users do not even have to commit to homogeneity. Furthermore, this approach allows for the development and use of ontologies and knowledge bases under a more conservative, bottom-up development model as an alternative to committing to a large, broad knowledge base. Thus, an alternative use of this model aims for the evolution of smaller, topic-specific ontologies intended for sharing by specialized communities. (Over time, it is possible that these topic-specific ontologies would grow and merge with others, so that an eventual end result might be a broadly encompassing ontology such as that sought by the Cyc effort.)

Our vision owes a great deal to Mark Stefik's (1986) view of knowledge bases as the next mechanism of information exchange. Stefik was interested, as are we, in both how systems of the future will be built and what uses will be made of them. He suggested that expert systems were parts and precursors of a new knowledge medium, a set of interactive multidisciplinary tools for information manipulation:

> A knowledge medium based on AI technology is part of a continuum. Books and other passive media can simply store knowledge. At the other end of the spectrum are expert systems which can store and also apply knowledge. In between are a number of hybrid systems in which the knowledge processing is done mostly by people. There are many opportunities for establishing human-machine partnerships and for automating tasks incrementally.

In these interactive tools, the computer provides storage, retrieval, and group communications services as well as languages and tools that enable precision and explicitness in manipulating knowledge about a topic.

In producing the new knowledge medium, Stefik argued that expert systems should be viewed as complex artifacts, such as automobiles or airplanes, which are assembled from highly sophisticated materials and subassemblies. A marketplace that supports the specialized suppliers of component technologies benefits the manufacturer of the end product. The marketplace provides economies of scale that make it possible to develop component technologies of a quality and sophistication that would be unaffordable if the costs had to be borne entirely within a single, built-from-scratch, one-of-a-kind product development effort. Thus, by analogy, Stefik concluded that

> ...the "goods" of a knowledge market are elements of knowledge... To reduce the cost of building expert systems, we need to be able to build them using knowledge acquired from a marketplace. This requires setting some processes in place and making some technical advances.

In particular, Stefik urged combining work on expert system shells with work on standard vocabularies and ways of defining things in terms of primitives. This suggestion is similar to the notion of ontologies proposed in this article. However, Stefik questioned (as do we) the feasibility of relying entirely on the development of standard vocabularies or ontologies for domains. The key to their effectiveness lies in how these ontologies are analyzed, combined, and integrated to create large applications. Tools and methods are needed to support this process. Our vision seeks to extend Stefik's by trying to further define the process as well as the supporting tools and methods.

Our particular extensions also have some kinship to the architecture of the national information infrastructure envisioned by Kahn and Cerf (1988) for the Digital Library System. Their vision of the library of the future consists of information collections bearing some analogies to Stefik's knowledge media. The Digital Library System provides a distributed network architecture for accessing information. The architecture contains database servers, various accounting and billing servers, servers to support placing knowledge into the library and extracting knowledge from it, and servers for translating knowledge flowing to and from the library into different forms. The Digital Library System suggests a model for how our vision of development, distribution, and dissemination of knowledge-based systems might be realized in the future. At the same time, our vision proposes supporting technology—for example,

There are many unanswered questions about how large-scale systems can best be built.

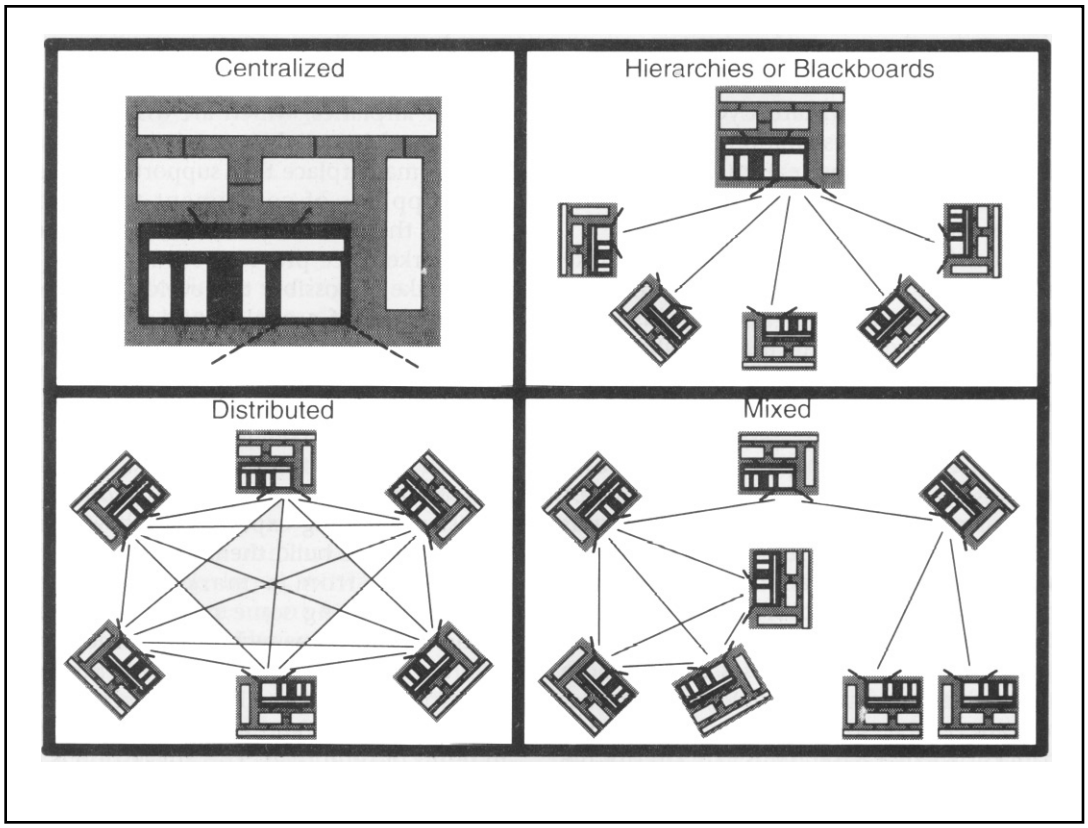


Figure 6. A Range of (potentially heterogeneous) System Models.

The technology described in this article enables experimentation with alternative models for architectures of knowledge-based systems. Shared ontologies provide the basis for protocols that link separate modules.

representation systems and translators— that could help realize the Digital Library System.

Supporting a Range of System Architectures

The technology described in this article is intended to provide enabling tools for building a range of architectures for large, knowledge-based systems. Figure 6 illustrates one dimension of this range. Within the scope of our vision, a system might be built as a single, large, integrated, centralized package (as depicted in figure 4). In this model, sharing and reuse occur at design time. Software-engineering concerns such as modularity can be achieved by partitioning the knowledge in this system into multiple bases linked by shared ontologies. Alternatively, however, the large system could be factored into multiple separate modules, communicating according to any of a number of alternative control regimes. Sharing and reuse in this fashion occur at run time. Modularity for software-engineering reasons can be handled by the separation of modules as well as by the same internal structuring, as in the first case. Some control regimes seemingly require that each module know a great deal about the others; the duplication of information in each module that is entailed is usually regarded as a potential maintenance bottleneck. However, the architecture provides natural mechanisms (ontologies and standard communications components) for encapsulating the knowledge needed for intermodule communication.

Over and above module structuring, system models can vary along a number of other dimensions in this scheme. These dimensions include the choice of representation languages used within modules, the homogeneity or heterogeneity of representation systems across modules, the use of specialized reasoning modules, the nature of ontologies, the content of the knowledge bases, the partitioning of the knowledge bases, the tightness of coupling with databases, the degree of transparency of modules (black boxes versus glass

boxes), and the locus of control.

There are many unanswered questions about how large-scale systems can best be built. Although sharing and reuse are clearly essential principles, the best way to make them operational remains to be understood. What are the trade-offs between sharing services at run time versus sharing knowledge bases at design time? On what scale and under what circumstances is translation viable? When are shared formalisms, rather than translation, required? How do domain-specific considerations drive choices of system models? What are the constraints on the usability of knowledge originally recorded for different purposes? What mechanisms best facilitate convergence between multiple users on a mutually satisfactory representation?

The intent behind the framework we outlined is not to enshrine a particular set of answers to such questions. Rather, our goal is to identify enabling technologies that make it easier to search the space of possible answers. The right answers will emerge only if we first make it easier for the AI community to explore the alternatives empirically—by building a number of large systems.

Working Groups in the Knowledge-Sharing Initiative

The desire to collaborate through knowledge sharing and reuse has arisen within a segment of the broad knowledge representation community that is interested in scaling up to larger systems and that views the sharing and reuse of knowledge bases as a means to this end. Closely related to this effort is a concern for building embedded systems in which knowledge representation systems support certain functions rather than act as ends in themselves.

In particular, our goal is to support researchers in areas requiring systems bigger than a single person can build. These areas include engineering and design domains, logistics and planning domains, and various integrated modality areas (for example, multimedia interfaces). Researchers working on such topics need large knowledge bases that model complex objects; because these models drive complex systems, they cannot be skeletons. Putting together much larger systems, of which various stand-alone systems being built today are just components, is an interesting challenge.

The creation of such knowledge resources requires communitywide effort. This effort engenders a need for agreed-on conventions to enable us to build pieces that fit together. Eventually, in pursuing the goal of large, shared knowledge bases as part of a nationwide information infrastructure, these conventions might become objects of study for the definition of more formal standards. (For those interested in the role of standards in computing infrastructure, Cargill [1989] is a useful entry point into the topic.) Currently, however, the conventions are intended to support experiments in knowledge sharing among interested parties.

The creation of… knowledge resources requires community-wide effort.

The next part of this article focuses on making our vision operational by developing these conventions. Doing so is an essential precursor to larger aspects of our vision, such as libraries of ontologies, reasoning modules, and representation systems. This section describes the activities of our four working groups on these foundation-laying activities. For each group, we summarize the problem being addressed, the approach being taken, and the outcomes sought.

Interlingua

The Interlingua Working Group is headed by Richard Fikes and Mike Genesereth, both of Stanford University.

Problem Formulation. The Interlingua Working Group focuses on the problems posed by the heterogeneity of knowledge representation languages. Specifically, to interchange knowledge among disparate programs (written by different programmers, at different times, in different languages), effective means need to be developed for translating knowledge bases from one specialized representation language into another. The goal of this group is to specify a language for communicating knowledge between computer programs (as opposed to a language for the internal representation of knowledge within computer programs). This language needs (1) an agreed-on declarative semantics that is independent of any given interpreter, (2) sufficient expressive power to represent the declarative knowledge contained in typical application system knowledge bases, and (3) a structure that enables semiautomated translation into and out of typical representation languages.

Approach. This group is specifying a language (KIF [knowledge interchange format]) that is a form of predicate calculus extended to include facilities for defining terms, representing knowledge about knowledge, reifying functions and relations, specifying sets, and encoding commonly used nonmonotonic reasoning policies. The group is also conducting knowledge-interchange experiments to substantially test the viability and adequacy of

The goal… is to specify a language for communicating knowledge between computer programs…

the language. The experiments focus on developing and testing a methodology for semiautomatic translation to and from typical representation languages and the use of the interchange format as an intermodule communication language to facilitate interoperability.

Outcomes. The specification for interlingua will evolve in a set of layers. The innermost layer will be a core, analogous to the primitives in Lisp, providing basic representational and language extension functions. The next layer will provide idioms and extensions that make the language more usable, analogous to the set of functions provided by Common Lisp. This working group will be responsible for developing these specifications. Its output will be (1) a living document containing the current KIF specification, describing open issues, and presenting current proposals for modification and (2) a corpus of documented *microexamples*, using fragments of knowledge bases to illustrate how they translate into KIF and to point out open issues.

By the time this group has completed its work, we expect that the documented interlingua specification will define a language in which the sharing and reuse of the contents of individual knowledge bases can be accomplished by transmitting specifications using KIF as the medium for expressing these languages. The language will be oriented toward supporting translation performed with a human in the loop, and we expect that several prototype translation aids will be developed during the course of this work.

Knowledge Representation System Specifications

The KRSS working group is headed by Bill Swartout of University of Southern California/Information Sciences Institute (USC/ISI) and Peter Patel-Schneider of AT&T Bell Labs.

Problem Formulation. This group is concerned with specification on a separate family-by-family basis of the common elements within individual families of knowledge representation system paradigms. The intent is not to develop a "be-all, end-all" knowledge representation system. The group is not trying to develop one language that encompasses all approaches to knowledge representation; rather, it seeks to create specifications that incorporate the good features within families and develop a common version that is reasonably comprehensive yet pragmatic for each individual family. An example of one such family is the set of KL-One descendents, that is, object-centered languages with definitional constructs, such as CLASSIC, LOOM, BACK, SB-One, and OMEGA. The effort should be viewed more as an attempt to develop a Common Lisp rather than a PL-1 or Ada. The analogy to Common Lisp is imperfect, however. Knowledge representation systems perform inference on the information represented within them; programming languages do not. Thus, specifying what inferences should be performed is an additional issue in specifying a knowledge representation system.

Approach. The goal of this group is to allow system builders working within a family to provide a common set of features that have consensus within this paradigm in addition to the augmentations that they regard as their research contribution. The resulting language will serve as a medium for sharing knowledge bases as well as a means for communicating ideas and issues among researchers. Potential benefits are the abilities of users to more readily convert between systems and to borrow models originally built to run in other systems. The trade-offs assumed by this group are the mirror image of those faced by the Interlingua Working Group: They eliminate the problem of translating knowledge bases between systems but require one to work within a given formalism to obtain this benefit.

Specifying a knowledge representation system poses the interesting challenge of specifying just what kinds of inference the system should perform. From Brachman and Levesque (1984), we know that if the system is expressive enough to be useful, the inferences that its reasoner draws will be incomplete. How should this incomplete inference be specified? The group's approach is to construct a layered specification. There will be an inner core of the language, which will consist

of a few constructs. Because this inner core will have limited expressivity, it will be possible to perform complete inference on it. Another layer, the outer core, will be built on the inner core. This layer will significantly extend the expressivity of the language, but inference on it will be incomplete. To specify the inferences that should be performed, the group will provide an abstract algorithm for drawing just those inferences that are required by the specification. Implementers of the specification must provide a reasoner at least as powerful as the one specified by this algorithm.

Outcomes. The group is seeking to develop a published specification and at least one initial implementation of a common language.

External Interfaces

The External Interfaces Working Group, cochaired by Tim Finin of the Unisys Center for Advanced Information Technology and Gio Wiederhold of Stanford University, focuses on interfaces that provide interoperability between a knowledge representation system and other software systems.

Problem Formulation. The time is ending when an intelligent system can exist as a single, monolithic program that provides all the functions necessary to do a complex task. Intelligent systems will be used and deployed in environments that require them to interact with a complex of other software components. These components will include conventional software modules, operating system functions, and database servers as well as other intelligent agents. There is a strong need to develop standard interface modules and protocols to make it easier to achieve this interoperability. The working group is concerned with three aspects of this problem: providing interoperability with other intelligent agents, conventional (for example, relational) database management systems, and object-oriented database systems.

Approach. To provide run-time interoperability between knowledge representation systems, we need a language or protocol that allows one system to pose queries or provide data to another. The group has begun the specification of such a language, KQML. The intent is that KQML will be to knowledge representation systems what SQL has become to database management systems—a high-level, portable protocol for which all systems will provide interfaces. The current specification is organized as a protocol stack in which the lowest information-conveying layer is based on the interlingua. Higher layers in this stack provide for modality (for example, assert, retract, query), transmission (for example, the specification of the recipient agent or agents), and complex transactions (for example, the efficient transmission of a block of data).

The integration of AI and database management system technologies promises to play a significant role in shaping the future of computing. As noted by Brodie (1988), this integration is crucial not only for next-generation computing but also for the continued development of database management system technology and the effective application of much of AI technology. The need exists for (1) access to large amounts of existing shared data for knowledge processing, (2) the efficient management of data as well as knowledge, and (3) the intelligent processing of data. The working group is studying the many existing interfaces between knowledge representation systems and relational databases (for example, Mckay, Finin, and O'Hare [1990]) and is attempting to develop specifications for a common one. The primary issues here are the various ways in which the data in the databases can best be mapped into the knowledge representation objects.

The third task that the group is looking at is providing interfaces between knowledge representation systems and object-oriented databases. The goal here is to be able to use an object-oriented database as a substrate under the knowledge representation system to provide a persistent object store for knowledge base objects. This work is exploratory, but the potential benefits in the long run are significant. They include (1) building and managing knowledge bases much larger than the current technology will support and (2) providing controls for transactions and concurrent access to knowledge bases at an object level.

Outcomes. The External Interfaces Working Group is concentrating on the development of the KQML protocol as its first goal. It hopes that an early implementation will be used to help build test beds for several distributed, cooperative demonstration systems. With regard to database interfaces, several working group members are attempting to integrate existing models for interfaces between knowledge representation systems and relational databases and to produce a specification of a common one. The working group is also planning an experiment in which a simple interface will be built to allow an existing object-oriented database to be used as a substrate under one of the representation systems being investigated by the KRSS working

… the goals proposed in this article suggest a number of high-payoff research questions for the entire research community.

group. This approach will provide a feasibility test and generate data for further exploration.

Shared, Reusable Knowledge Bases

The Shared, Reusable Knowledge Bases Group is headed by Tom Gruber of Stanford University and Marty Tenenbaum of EITech, Inc.

Problem Formulation. This group is working on mechanisms to enable the development of libraries of shareable knowledge modules and the reuse of their knowledge-level contents. Today's knowledge bases are structured as monolithic networks of highly interconnected symbols, designed for specific tasks in narrow domains. As a result, it is difficult to adapt existing knowledge bases to new uses or even to identify the shareable contents. To enable the accumulation of shareable knowledge and the use of this knowledge by multiple systems, we need a means for designing composable modules of knowledge. The working group is chartered to identify the barriers to the building and use of shared knowledge modules, characterize potential approaches to overcoming these barriers, and conduct experiments exploring mechanisms for knowledge sharing.

Approach. The working group supports three kinds of activity. One is the identification of important research issues for knowledge sharing, including problems of methodology (for example, multidisciplinary, collaborative knowledge base design) as well as engineering (for example, scalability, shareability). A second activity is the development of ontologies that define terminology used to represent bodies of shareable knowledge. The task includes (1) identifying bodies of knowledge worth the effort to formally represent and make shareable and (2) defining coherent sets of terms that characterize the ontological commitments and representational choices for modeling these bodies of knowledge. A third type of working group activity is the coordination of collaborative experiments in knowledge sharing in which multiple research groups attempt to share and use each other's knowledge bases (for example, libraries of device models). Some experiments will evaluate the use of ontologies as a mechanism for sharing (that is, for modularity and composability of knowledge modules and the specification of their contents).

Outcomes. To these ends, the working group is concentrating on three objectives. The first is a survey of the state of the art in research on knowledge sharing and reuse, which identifies the techniques currently being explored and recommends research on the critical problems to be solved. A second outcome is a set of results from the collaborative experiments on knowledge sharing, including the ontologies used for each experiment and lessons learned about tools and methodologies for developing them. An immediate subgoal for this outcome is to develop a mechanism for representing these ontologies in portable form, building on the work of the other three working groups. The third, more long-term objective is to develop a suite of exemplary shared ontologies and the knowledge bases using them. These ontologies will serve as research test beds, playing a role analogous to the E. *coli* bacterium in biology: a well-understood, complete example on which to perform a variety of experiments.

Long-Term Research Issues

The preceding description of the working groups focused on near-term issues. However, the goals proposed in this article suggest a number of high-payoff research questions for the entire research community. In this section, we want to focus on longer-term concerns by reviewing some of these questions.

A number of issues raised by this vision were also identified by Ron Brachman (1990) in his AAAI-90 address on the future of knowledge representation. High on his list of issues were knowledge representation services, knowledge base management, and the usability of knowledge representation systems. Brachman pointed out that the idea of a single, general-purpose knowledge representation system with optimal expressive power

might not meet real needs. Instead, he urged, we must look at different levels of service and understand what it means to offer them, how to characterize the capabilities of different services, and what the cost is of invoking them. The management of very large knowledge bases poses some fascinating research questions, including methods for handling globally inconsistent but locally reasonable knowledge, higher-level principles for organizing knowledge, and techniques for belief revision as knowledge bases evolve over time. As large knowledge bases come into widespread use, they will need to be built, extended, and used by a wider range of users. Because these users will likely be less forgiving than academic researchers using products of their own making, a number of questions arise concerning presenting knowledge, browsing, querying, and explaining.

Other relevant issues raised by Brachman include the integration of multiple paradigms, extensible representation systems, and efficiency concerns. In addition, each of the four working groups has questions that present challenges to the research community.

The notion of translation implied by interlingua raises a number of questions about tools and methodologies for translation. As the work on interlingua has progressed, a much better understanding has grown about the distinction between communication codes and representation codes (these two terms were introduced into the discussion by Pat Hayes). A *communication code* captures knowledge for the purposes of exchanging it or talking about it, and a *representation code* attempts to capture knowledge for the purpose of supporting the efficient implementation of inference mechanisms. As the effort proceeds, it is likely to spawn a great deal of work on understanding and recording the design rationale behind representation codes to facilitate greater automation in translating them into and out of communication codes.

A number of issues in representation languages remain as research topics. Belief has already been mentioned. Others include defaults and inheritance, negation, disjunction, metadescriptions, higher-order logics, description of inference, predicate reification, causality, and time.

External interfaces present a range of both short- and long-term research issues. Examples include the verification of protocols for asynchronous operation of multiple end user applications with many knowledge bases, the assessment of possible degrees of parallelism, and deeper investigation into requirements for supporting interactive user interfaces. Longer-range issues include mechanisms for specifying the quantity and quality of information to be returned by a knowledge base in response to a request, means for dealing with uncertainty, and methods for optimizing the performance of distributed systems.

Finally, the notion of shared, reusable knowledge bases provides a tremendous amount of grist for the research mill. Most obviously, researchers will be exploring questions about the content of specific reusable ontologies for some time to come. In addition, there are a number of ancillary questions: How is consensus on a group ontology best achieved? How can consensus be maintained as needs change over time? What kinds of automated assistance and interactive tools will be beneficial? How can we verify compatibility with an ontology? How can we support correspondence theories that allow knowledge base developers to express and reason about mappings between different ontologies?

The efforts described in this article to develop conventions for sharing and reuse represent the start—rather than the culmination—of a large body of research activity. We believe that this area is one in which conventions will serve to focus and enable research, not codify and terminate it.

Conclusion

Attempting to move beyond the capabilities of current knowledge-based systems mandates knowledge bases that are substantially larger than those we have today. However, representing and acquiring knowledge is a difficult and time-consuming task. Knowledge-acquisition

The time is ripe to start building the infrastructure for integrating AI software at the knowledge level.

tools and current development methodologies will not make this problem go away because the root of the problem is that knowledge is inherently complex and the task of capturing it is correspondingly complex. Thus, we cannot afford to waste whatever knowledge we do succeed in acquiring. We will be hard pressed to make knowledge bases much bigger than we have today if we continue to start from scratch each time we construct a new system. Building qualitatively bigger knowledge-based systems will be possible only when we are able to share our knowledge and build on each other's labor and experience.

The time is ripe to start building the infrastructure for integrating AI software at the knowledge level, independent of particular implementations. Today, there is a significant body of ongoing work in AI and application domains on pieces of the problem, such as basic knowledge representation; knowledge-acquisition tools; task-specific architectures; and domain-oriented, multiuse models. What is lacking is an integrating framework, the means for describing, connecting, and reusing knowledge-based technology.

Addressing these concerns will open the doors to the development of much larger-scale systems, structured in a fashion that facilitates their development, maintenance, and extension. Furthermore, it will eliminate barriers to embedding AI components in larger, mostly conventional software systems. This approach will lead to a great expansion in the range of applicability for AI technology, which, in turn, will greatly enhance its utility and significantly expand the commercial marketplace.

The knowledge representation technology that supports these goals will have four key characteristics:

First, it will offer libraries of reusable ontologies, that is, knowledge base frameworks consisting of (1) formal definitions of the terms that can be used to model a domain or class of domains and (2) assertions that govern the requirements and constraints for creating valid models within a domain by combining and relating terms and term instances.

Second, it will offer powerful, expressive, and efficient interpreters and compilers for knowledge representation systems (knowledge bases combined with inference mechanisms) in which these ontologies are embedded. These systems will likely be structured as services oriented toward a client-server model of interaction.

Third, it will provide system builders with tools for translating between alternative representation systems. These tools will enable them to create efficient, optimized systems by making choices about alternative implementations without losing the opportunity to reuse representational work from other, different formalisms.

Fourth, it will embed these interpreters and compilers in architectures that support complete interoperability not just between multiple knowledge-based systems but also with conventional software systems. In particular, there will be a convenient, standard application programming interface and tight interconnection with databases.

This article attempted to articulate a vision of the necessary knowledge representation system technology and a path to achieving it. It also argued that progress in this area will dramatically change for the better the way that knowledge-based systems are built and the way they are perceived by their users. Central to this vision is the notion of establishing an information infrastructure for promoting the sharing and reuse of knowledge bases in the development and application of large, enterprise-level software systems.

Acknowledgments

This effort is supported by NSF grant IRI-900-6923, NASA-Ames contract NCC 2-520, and a cooperative agreement between USC/ISI and the Corporation for National Research Initiatives. Steve Cross, Mike Genesereth, Bob Kahn, Bill Mark, Ron Ohlander, Peter Patel-Schneider, Marty Tenenbaum, and Gio Wiederhold deserve particular acknowledgment for their respective important roles in shaping the efforts presented in this article. The authors would like to acknowledge beneficial discussions with Brad Allen, Giuseppe Attardi, R. Bhaskar, Danny Bobrow, B. Chandrasekaran, Randy Davis, Helen Gigley, Pat Hayes, Phil Hayes, Alan Lawson, Bob Mac Gregor, Elaine Rich, Luc Steels, Mark Stefik, Abe Waksman, Chuck Williams, and Mike Williams. Sheila Coyazo's copyediting assistance was extremely valuable.

References

Bennett, J. S. 1984. ROGET: Acquiring the Conceptual Structure of a Diagnostic Expert System. In Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems, 83–88. Washington, D.C.: IEEE Computer Society.

Brachman, R. J. 1990. The Future of Knowledge Representation. In Proceedings of the Eighth National Conference on Artificial Intelligence, 1082–1092. Menlo Park, Calif.: American Association for Artificial Intelligence.

Brachman, R. J., and Levesque, H. J. 1984. The Tractability of Subsumption in Frame-Based Description Languages. In Proceedings of the Third National Conference on Artificial Intelligence, 34–37. Menlo Park, Calif.: American Association for Artificial Intelligence.

Brodie, M. 1988. Future Intelligent Information Systems: AI and Database Technologies Working Together. In *Readings in Artificial Intelligence and Databases*, eds. J. Mylopoulos and M. Brodie, 623–641. San Mateo, Calif.: Morgan Kaufmann.

Cargill, C. F. 1989. *Information Technology Standardization: Theory, Process, and Organizations*. Bedford, Mass.: Digital.

Chandrasekaran, B. 1986. Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design. *IEEE Expert* 1(3): 23–30.

Chandrasekaran, B. 1983. Toward a Taxonomy of Problem-Solving Types. *AI Magazine* 4(4): 9–17.

Cutkosky, M. R., and Tenenbaum, J. M. 1990. A Methodology and Computational Framework for Concurrent Product and Process Design. *Mechanism and Machine Theory* 25(3): 365–381.

Feiner, S. K., and McKeown, K. R. 1990. Coordinating Text and Graphics in Explanation Generation. In Proceedings of the Eighth National Conference on Artificial Intelligence, 442–449. Menlo Park, Calif.: American Association for Artificial Intelligence.

Finin, T., and Fritzson, R. 1989. How to Serve Knowledge—Notes on the Design of a Knowledge Server. Presented at the AAAI Spring Symposium on Knowledge System Tools and Languages, 28–30 March, Stanford, Calif.

Forbus, K. 1990. The Qualitative Process Engine. In *Readings in Qualitative Reasoning about Physical Systems*, eds. D. Weld and J. de Kleer, 220–235. San Mateo, Calif.: Morgan Kaufmann.

Gruber, T. R. 1991. An Experiment in the Collaborative Development of Shared Ontology, Technical Report, KSL 91-51, Knowledge Systems Laboratory, Stanford University. Forthcoming.

Guha, R. V., and Lenat, D. B. 1990. CYC: A Mid-Term Report. *AI Magazine* 11(3): 32–59.

Harp, B.; Aberg, P.; Benjamin, D.; Neches, R.; and Szekely, P. 1991. DRAMA: An Application of a Logistics Shell. In Proceedings of the Annual Conference on Artificial Intelligence Applications for Military Logistics, 146-151. Williamsburg, Va.: American Defense Preparedness Association. Forthcoming.

Johnson, W. L., and Harris, D. R. 1990. Requirements Analysis Using ARIES: Themes and Examples. In Proceedings of the Fifth Annual Knowledge-Based Software Assistant Conference, 121–131. Liverpool, N.Y.: Rome Air Development Center.

Kahn, R. E., and Cerf, V. E. 1988. An Open Architecture for a Digital Library System and a Plan for Its Development, Volume I: The World of Knowbots, Technical Report, Corporation for National Research Initiatives, Reston, Virginia.

Kahn, G.; Nowlan, S.; and McDermott, J. 1984. A Foundation for Knowledge Acquisition. In Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems, 89–96. Washington, D.C.: IEEE Computer Society.

McDermott, J. 1990. Developing Software Is Like Talking to Eskimos about Snow. In Proceedings of the Eighth National Conference on Artificial Intelligence, 1130–1133. Menlo Park, Calif.: American Association for Artificial Intelligence.

Mckay, D.; Finin, T.; and O'Hare, A. 1990. The Intelligent Database Interface. In Proceedings of the Eighth National Conference on Artificial Intelligence, 677–684. Menlo Park, Calif.: American Association for Artificial Intelligence.

Pan, J. Y.-C.; Tenenbaum, J. M.; and Glicksman, J. 1989. A Framework for Knowledge-Based Computer-Integrated Manufacturing. *IEEE Transactions on Semiconductor Manufacturing* 2(2): 33–46.

Roth, S., and Mattis, J. 1990. Data Characterization for Intelligent Graphics Presentation. In Proceedings of CHI-90, The Annual Conference on Computer-Human Interaction, 193–200. New York: Association of Computing Machinery.

Sathi, A.; Fox, M. S.; and Greenberg, M. 1990. Representation of Activity Knowledge for Project Management. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-7(5): 531–552.

Smith, D. R. 1990. A Semiautomatic Program Development System. *IEEE Transactions on Software Engineering* SE-16(9): 1024–1043.

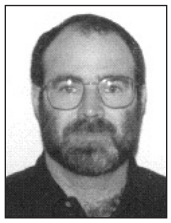
Steele, G. R. 1984. *Common Lisp: The Language*. Bedford, Mass.: Digital.

Steels, L. 1990. Components of Expertise. *AI Magazine* 11(2): 29–49.

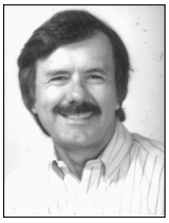
Stefik, M. 1986. The Next Knowledge Medium. *AI Magazine* 7(1): 34–46.

Swartout, W. R., and Smoliar, S. 1989. Explanation: A Source for Guidance in Knowledge Representation. In *Knowledge Representation and Organization in Machine Learning*, ed. K. Morik, 1–16. New York: Springer-Verlag.

Waterman, D. A. 1986. *A Guide to Expert Systems*. Reading, Mass.: Addison-Wesley.



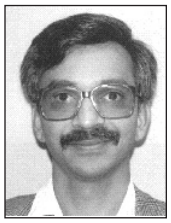
Robert Neches is head of the Integrated User-Support Environments Group at USC/Information Sciences Institute and a research associate professor in the Computer Science Department at USC. His current research topics include intelligent interfaces and development methodologies for large systems integrating AI and conventional technology. Since November 1989, he has been serving as coordinator for a university, government, and private industry effort to develop conventions that facilitate the sharing and reuse of AI knowledge bases and knowledge representation system technology.



Richard Fikes is a research professor in the Computer Science Department at Stanford University and is co-director of the Heuristic Programming Project in Stanford's Knowledge Systems Laboratory. Fikes is a fellow of the American Association for Artificial Intelligence and is prominently known as co-developer of the STRIPS automatic planning system and one of the principal architects of IntelliCorp's KEE system.

Tim Finin is professor and chairman of the Computer Science Department at the University of Maryland. Prior to joining the University of Maryland, he was a technical director at the Unisys Center for Advanced Information Technology, a member of the faculty of the University of Pennsylvania, and on the research staff of the Massachusetts Institute of Technology AI Laboratory. He has had over 20 years of experience in the applications of AI to problems in database and knowledge base systems, expert systems, natural language processing, and intelligent interfaces. Finin holds a B.S. degree in electrical engineering from the Massachusetts Institute of Technology and a Ph.D in computer science from the University of Illinois.

Thomas Gruber is a research scientist in the Knowledge Systems Laboratory, Stanford University. His current research interests include the development of shared ontologies for knowledge sharing and reuse, the acquisition and generation of design rationale, and the computer-assisted formulation of engineering models. His earlier work at the University of Massachusetts was in the areas of knowledge-based communication prosthesis and automated knowledge acquisition for expert systems.



Ramesh Patil recently joined the Intelligent Systems Division of USC/ Information Sciences Institute as a project leader after serving as an associate professor of computer science at MIT. His Ph.D. work on multilevel causal reasoning systems for medical diagnosis, performed in the Computer Science Department at MIT, was the basis for his receiving the American Association for Medical Systems and Informatics Young Investigator Award in Medical Knowledge Systems. His current research interests include knowledge representation, signal interpretation, qualitative reasoning, and diagnostic reasoning in medicine and electronics.

Ted Senator is chief of the Artificial Intelligence Division in the Financial Crimes Enforcement Network of the U.S. Treasury Department. In his prior position with the Department of the Navy, he managed the Navy's applications under the Strategic Computing Initiative.

William R. Swartout is director of the Intelligent Systems Division and an associate research professor of computer science at USC/Information Sciences Institute. He is primarily known for his work on explanation and new expert system architectures. Swartout was co–program chair for the 1990 National Conference on Artificial Intelligence and is currently serving on the Board of Councilors of the American Association of Artificial Intelligence.