Mobile agents can benefit from standards efforts on interagent communication

Tim Finin, Yannis Labrou and Yun Peng[1]

Department of Computer Science and Electrical Engineering

University of Maryland, Baltimore County

Baltimore, MD 21250

ABSTRACT

On the road for the future success of mobile agents, we believe that inter-agent communication is an issue that has not been adequately addressed by the mobile agents community. Supplementing mobile agents with the ability to interact with other mobile or static agents, or agentified information sources is a necessity in the vastly heterogeneous arena where mobile agents are called to compete. Thus, an agent communication language should be interpreted as a tool with the capacity to integrate disparate sources of information. In the first segment, we argue that mobile agents can benefit from current standards efforts on agent communication since the focus of such work is to address heterogeneity by defining a "common language" for communicating agents. In the second part, we discuss ongoing research on agent to agent communication and we present current standards efforts relevant to agent communication.

[1] Final draft of the paper which appears in IEEE Communications Magazine, v36 n8, July 1998, pp. 50-56.

Introduction

Imagine that some huge calamity has befallen you. You have to choose between losing your ability to communicate (in any form) and losing your ability to move around. Which one of the two evils would you choose? Granted, being confined in a single space can be particularly disagreeable; incarceration is probably the most common form of that state. On the other hand, consider what losing your ability to communicate might entail. Imagine that you are faced with the simple task of mailing a package to a relative. You can go to the post office but since you can not communicate the task at hand, you have to master your way through the postal service and eventually take care of sending this package all by yourself. There are occasions of course, where nothing beats performing a task by yourself but more often than not, someone else is much more of an expert with a particular task than you are. Then, being able to describe the task and communicate your desire for it to be performed is essential.

Notice that we are referring to communication without necessarily implying verbal communication in a spoken language. What is essential is that this language is common and shared, i.e., its "words" have the same meaning for both communicating parties. From English, or Greek, or Mandarin Chinese to sign language for the deaf, or even the signs that baseball managers, coaches and players use, the indispensable property of a useful "language" is that the meaning of its tokens is shared, thus allowing for communication between willing participants. We argue, then, that communication usually means communication in a "common language", which in turn requires a shared, mutual understanding of the vocabulary of that language. Of course, "knowing" a language, i.e., knowing the meaning of its tokens is different from knowing how to communicate in it, i.e., knowing how to use its tokens in order to achieve goals and effect others and one's surroundings.

Let us explore these observations in the universe of software agents. Software agents are an emerging software-building paradigm. Leaving the hype aside for a moment, the paradigm introduces a powerful and ubiquitous abstraction that exhibits the following (among others) key concepts: (a) a software agent is an autonomous goal-directed process capable of performing actions, (b) is situated in, is aware of and reacts to its environment, and (c) cooperates with other agents (software or human) to accomplish its tasks. Mobile agents may be thought as programming entities that can freely roam the network and act on behalf of their users [1]. A slightly more technical definition suggests that "mobile agents are programs, typically written in a script language, which may be dispatched from a client computer and transported to a remote server computer for execution" [2]. From a historical perspective, the work on distributed computing led to mobile agents following a route through the problem of process migration and the concept of mobile objects [1].

The distinguishing characteristic, in terms of being "agents", of such "migrating processes" or "mobile objects" is not their mobility, *i.e.*, their ability to move around the network carrying their execution state and/or data, but their ability to act (autonomously) on behalf of their users and eventually perform tasks for them. Mobility is orthogonal to communication; what is important is that both functionalities are ways to perform tasks. We will forego the potentially endless process of arguing whether one of the two might be subsumed by the other and we are willing to settle for the reconciliatory viewpoint that both abilities have their place in an agent's life. But, as we will argue, the ability to communicate is more essential than mobility for a piece of code that aspires to be called an agent, in the sense that an agent can afford to be non-mobile but by being non-communicative, the agent risks isolation and might eventually face inability to perform its tasks.

Communication between agents, we suggest, is not about transporting bits across a communication channel but a means for achieving high-level interoperability (via communication), at a higher-level of abstraction, which involves such concepts as beliefs, goals, expectations and intentions. Standardization efforts have been concerned with capturing what "communication in a common language" might mean for software agents, with the understanding that addressing this question will lead to the building of software agents that will be true to their name. In the first logical part of our presentation, we discuss the problem of interoperability for mobile agents and we relate it to the broader interoperability issues and methodologies that concern the wider agent community. In the second part, we introduce the powerful notion of an Agent Communication Language (ACL); an ACL introduces a versatile interoperability approach.

Standardization Efforts: in Search of a Common Language for Agents

Vast amounts of ink and bandwidth have been consumed over the past few years in attempts to provide a definition of a software agent. Fruitless as such intellectual exercises might have been with respect to providing a crisp definition or even a consensus working definition, they have been useful in terms of exploring the possible taxonomies of agents [3-5]. One of the conclusions that has arisen from these discussions is that software agents define more of a paradigm than an artifact. This paradigm is a continuously evolving one. But despite the state of constant flux that the field experiences, the standardization efforts that are of interest to the agents community are centered on the problem of interoperability between agent applications. Unfortunately, an evolving paradigm is not what standards efforts usually have as their scope. Standards efforts are guided by a specific task; software agents merely introduce a way of developing code for a multitude of tasks. Software agents can benefit from standards, but the question is what kinds of standards? Moreover, if the subject matter of the standards efforts is a moving target, towards what are the standards going to apply?

Before answering these questions, let us point out some of the realities that standards efforts ought to take into account: (a) various languages, representing different programming paradigms

(object-oriented, logic, functional, *etc.*) will be used for implementing agents, (b) hardware platforms and operating systems are expected to be equally varied, and (c) agents are going to be written as autonomous applications and thus few assumptions might be made about their internal structure. Before such efforts are crushed under the weight of their own high aspirations, such as making any two pieces of software seamlessly integrate with one another, we should consider re-orienting their scope.

So, the answer to the "what kinds of standards" question is that standards efforts in the area of software agents emphasize the identification of distinct layers towards which standardization principles may be applied.

- One layer is that of translation between languages in the same family (or between families) of languages. This is a very formidable task. The Object Management Group (OMG) standardization effort is an example of work in this direction, within the family of object-oriented languages.

- Another layer is concerned with guaranteeing that the semantic content of tokens is preserved among applications; in other words, the same concept, object, or entity has a uniform meaning across applications even if different "names" are used to refer to it. Every application incorporates some view of the domain (and the domain knowledge) it applies to. The technical term for this body of "background" knowledge is "ontology." More formally, an ontology is a particular conceptualization of a set of objects, concepts and other entities about which knowledge is expressed and of the relationships that hold among them. An ontology consists of terms, their definitions, and axioms relating them [6] ; terms are normally organized in a taxonomy. For example, the fact that a book can be described with attributes such as its title, its ISBN and its author(s) name(s) is a particular conceptualization of the concept "book." This conceptualization exists and makes sense regardless of the presence of an application which employs it or of the particular strings one uses to "encode" the "title", "ISBN#" and "author" attributes.

- A third layer relates to the communication between agents. This is not about transporting bits and bytes between agents; agents should be able to communicate complex "attitudes" about their information and knowledge content. Agents need to ask other agents, to inform them, to request their services for a task, to find other agents who can assist them, to monitor values and objects, and so on. Such functionality, in an open environment, can not be provided by a simple Remote Procedure Call (RPC) mechanism.

The fate of such standards' efforts is less certain than their persistent focus. Although major corporations take a close interest to these initiatives and participate actively in them, alongside the academia, the explorative nature of this work limits the possibility of an immediate industry-wide adoption. The intent can be described as an attempt to provide solutions and approaches that interested parties will gradually adopt because they facilitate their task and not because some body

sanctions them or because there is an industry-wide agreement to adopt them. The stakes might be too high for such agreements.

Interoperability Problems for Mobile Agents

Mobile agents reside in a highly heterogeneous environment; this heterogeneity presents itself in many dimensions. Mobile agents migrate to a host where an execution environment is set up for them; upon arriving there, they might execute code, make Remote Procedure Calls (RPCs) in order to access the resources of the host, collect data and eventually might initiate another process of migration to another host. While residing on a particular host, mobile agents might have limited interaction (communication) with other agents on the host through an RPC-type mechanism. Such hosts, *i.e.*, the execution environments, come with many names, but we will use the term "mobile agent platform" to refer to them. A potential problem arises from the fact that not all platforms for mobile agents are the same. The Mobile Agent Facility (MAF) proposal [7] is an attempt to standardize this execution environment. The proposal was still under investigation by the Object Management Group (OMG), as of the end of 1997. Although MAF has gained some momentum within the community, there seems to be quite a variety of mobile agents' platforms coming from both the industry and the academia. We do not expect these differences to be reconciled in the immediate future.

So, the first question to ask is what happens when a mobile agent needs to interact with mobile agents that are currently hosted in a platform of a different design; in such a case a mobile agent can not even visit such a platform. Also, let us consider the case of a mobile agent that needs to interact with some agent that is not endowed with mobility. In this case, there is no agent platform to visit. Should the mobile agent be prevented from interacting with the static agent? Yet another awkward case might arise when a mobile agent needs to interact with an information source that is not a full-fledged agent but might offer some way of interacting with its information content. Nevertheless, regardless of these discrepancies, it will be necessary for all these disparate sources of information content to seamlessly interact with one another. Interaction means more than simply exchanging messages; it also involves facilities for finding these information sources. "Finding," means not only physically locating them on the network, but also been able to identify them or judge them relevant based on their apparent content or capabilities.

A whole different (qualitatively) range of problems appears at a different level. These potential conversants for the mobile agents, *i.e.*, mobile agents implemented for different platforms, static agents and non-agent information sources, may be implemented or make use of languages different from the mobile agent's one. Furthermore, the tokens they use to refer to the same concepts, entities and objects might differ; one agent might refer to the title of a book as "book_title" and another might use the token "title", but we would like to think of both as referring to the same

concept. To address all of these issues the community has suggested a layered abstraction that treats agents at a higher level than the details of their internal structure.

Mobile Agents: A Common Agent Communication Language Means Interoperability

Three basic problems need to be addressed for agents to communicate in a common language. First, how we can translate from one language to another; second, how we can guarantee that the meaning of concepts, objects, relationships is the same across different agents; and third, how this potentially sharable knowledge is going to actually be shared, communicated between agents. An Agent Communication Language (ACL) is a tool that follows the path of this layered abstraction of the interoperability question.

An agent communication language can be thought of as a collection of message types each with a reserved meaning. A communication language is not concerned with the physical exchange, over the network, of an expression in some language, but rather with stating an "attitude"[2] about the content of this exchange. Therefore, for example a communication language can distinguish between a particular message exchange that suggests a query, an interest for a particular content, or an assertion. From a software engineering point of view, an agent communication language can be viewed as another messaging protocol, but with two major differences: (a) it describes the application and the actions it can perform, or it can be requested to perform, at a higher level of abstraction, and (b) offers a larger variety of message types. An ACL introduces a powerful abstraction because it separates (1) the expressions that are the content of the exchange and (2) their meaning, from the attitude that is expressed about them.

An agent communication language can now be viewed under a different light. An ACL is an interoperability tool, or at least a conceptual framework that can assist us in addressing the "ugly" problem of interoperability between applications. Although interoperability is not a problem unique to agents, the very fact that software agents are meant to be, almost by definition, autonomous applications designed with minimal a priori expectations for the state of the rest of the universe brings interoperability to the forefront. Mobile agents are even more susceptible to the perils of heterogeneity. An agent communication language offers the following advantages:

- Generally speaking, it can be used as an interoperability tool

---

[2] We are using the concept of a "propositional attitude" from logicians here. This is a three part relationship among an agent (e.g., john), a content-bearing proposition (e.g., "it is raining") and one of a finite set of possible attitudes the agent might have with respect to the proposition (e.g., believing, wondering, hoping, asserting, fearing, etc.).

- between static and mobile agents, or
- between mobile agents designed for different agents' platforms, or finally,
- between mobile agents and "static" agentified information sources.

- An agent communication language complements the traditional procedural approach that dominates the mobile agents paradigm with the potential advantages of the declarative approach ("how to" perform a task vs. describing "which" task is to be performed).
- Finally, an agent communication language introduces a level of abstraction that can accommodate multiple paradigms.

Thus far, most of the work on mobile agents has focused on the problem of designing the agent platform and addressing major issues surrounding its design, such as security and authentication. Agent communication has been put in the backburner, partly because of the concentration on fundamental issues of mobility and partly because the ACL alludes to a declarative approach that at least in the surface, contradicts the procedural approach that existing mobile agent systems favor. The only exception to this trend is experimental work by the *Agent Tcl* group, in Dartmouth, which has tried to integrate the agent communication language KQML to a mobile agents framework [8].

Agent Communication Standards in Detail

We will discuss current standards efforts on agent communication. We argued that communication almost always means "communication in a common language." Granted, effective communication between software agents (or any other entities for that matter) requires more than just sharing a common language; knowledge of *how to* use some common language in order to achieve goals and effect one's environment is the purpose of possessing the understanding of any language. Ongoing standards efforts though, are concerned primarily with the first problem. *How to* use the common language is currently left up to the individual agents because: (a) differences at the complexity level of individual agents dictate different approaches, (b) domain-specific considerations lead to specialized behaviors, and (c) we lack the application-driven momentum that will force us to address such issues. From a standardization point of view, any work on the "how to" problem will have to rely on the solution(s) to the "common language" problem. At this time, as the "common language" efforts reach some maturity, researchers start to grapple with standardization issues of the *how to* problem. Mastering the vocabulary of a language precedes its effective use.

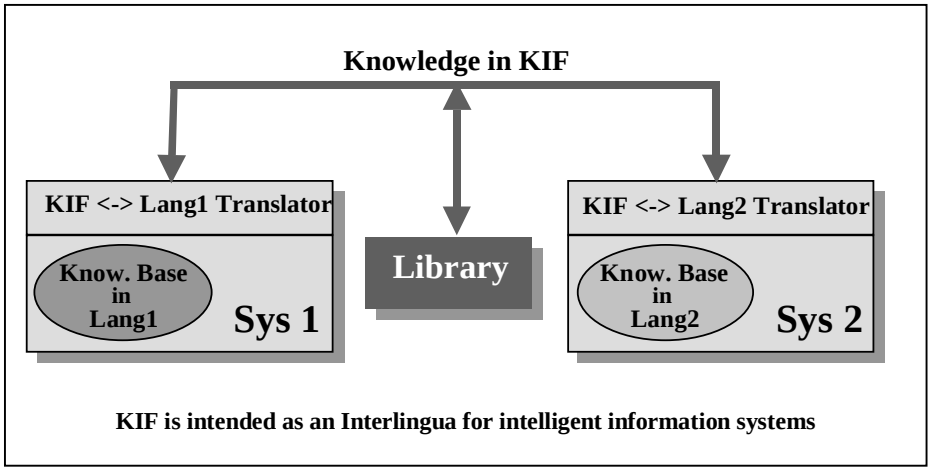
The Knowledge Sharing Effort (KSE) [9] [10] was an initiative that tried to attack the interoperability question along the lines of the three layers we presented. The KSE work resulted

into three languages, each addressing one of the layers: (1) the Knowledge Interchange Format (KIF) [11] language was designed as a solution to the translation problem, (2) Ontolingua was developed as a language in which one can write portable ontologies, and (3) Knowledge Query and Manipulation Language (KQML) was specified as a language for inter-agent communication. We will briefly review KIF, cover KQML in more detail and discuss the ontological question, which we believe to be the most common obstacle for agent interaction.

KIF as an Interlingua for Encoding Content

Specialized languages have been developed which are particularly good at describing certain fields. For example, STEP (Standard for the Exchange of Product Model Data) [12] is an ISO standards project working towards developing mechanisms for the representation and exchange of a computerized model of a product in a neutral form. SGML is an example of a language, which is designed to describe the logical structure of a document. There are special languages for describing workflow, processes, chemical reactions, etc.

Figure 1. KIF as an Interlingua



It would be nice if there were some expressive languages and related computer systems which were good at representing a very broad range of things, like the natural languages, but which do not suffer the problems of imprecision and ambiguity. Database systems and their languages (e.g., SQL, OQL) offer one general approach and certain object-oriented languages perhaps offer another. However, it is difficult or impossible to capture all kinds of information and knowledge in most of these general languages.

One of the general tools which mathematicians have developed for describing things is logic. Going back to Wittgenstein, many mathematicians, philosophers and computer scientists have tried to use symbolic logic to clearly and unambiguously define and describe things and situations of interest. Rather simple logics (e.g. the first order predicate calculus) turn out to be very power-

ful in terms of the kinds of things with which they can be used to describe. Some mathematicians and computer scientists (e.g., John McCarthy [13]) argue that a relatively simple logic is sufficient to express anything of interest or utility to people and other intelligent agents - simple concrete facts, definitions, abstractions, rules, constraints, meta-knowledge (knowledge about knowledge), etc.

KIF, a particular logic language, has been proposed as a standard to use to describe things within computer systems, e.g. expert systems, databases, intelligent agents, etc. Moreover, it was specifically designed to make it useful as an "interlingua". By this we mean a language which is useful as a mediator in the translation of other languages. For example, people have built translation programs that can map a STEP/PDES expression into an equivalent KIF expression and vice versa. If we had another language for electronic commerce, say MSEC, we could develop a translator to and from KIF. We would then have a way to translate between STEP/PDES and MSEC using KIF as an intermediate representation.

KIF is a prefix version of first order predicate calculus with extensions to support non-monotonic reasoning and definitions. The language description includes both a specification for its syntax and one for its semantics. First, KIF provides for the expression of simple data. For example, the sentences shown below encode tuples in a book-seller's database (arguments stand for ISBN, thematic category and price, respectively):

```
(book 015-46-3946 Architecture 72)

(book 026-40-9152 Fiction 36)
```

More complicated information can be expressed using complex terms. For example, the following sentence states that one book is more expensive than another:

```
(> (price book1) (price book2))
```

KIF includes a variety of logical operators to assist in the encoding of logical information (such as negation, disjunction, rules, quantified formulas, and so forth). The expression shown below is an example of a complex sentence in KIF. It asserts that the number obtained by raising any real number ?x to an even power ?n is positive:

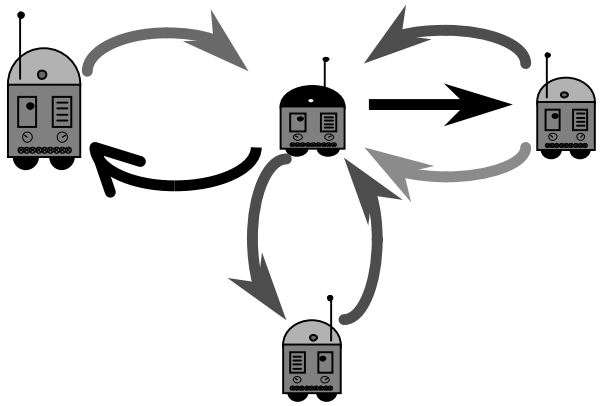
```
(=> (and (real-number ?x)  (even-number ?n))
    (> (expt ?x ?n) 0))
```

KQML as an Agent Communication Language

There are many approaches and efforts trying to address the ontological problem, but when it comes to agent communication, KQML almost monopolizes the debate. Recently though (1996), an international standards body was founded, called Foundation for Intelligent Physical Agents

(FIPA). FIPA is supported by major telecommunication companies and industrial partners and has as its goal the development of standards in the area of agents. FIPA has proposed an ACL (temporarily called FIPA ACL) that follows the path of KQML and tries to build on the lessons of the KQML experience. We will not cover FIPA ACL in more detail, but the interested reader

can find more about it through the FIPA homepage at http://drogo.cselt.stet.it/~fipa and the current versions of the FIPA specification documents. Whatever differences exist between the FIPA ACL and KQML lie at the semantic level; covering them would be beyond the scope of this presentation and would distract the reader from understanding the basic concepts behind an ACL. Our coverage of KQML should prepare the reader for FIPA ACL, since the two are similar in spirit,



syntax and objectives.

Figure 2. KQML-speaking agents, engaged in message exchange

KQML [14, 15] draws its approach from speech act theory [16, 17] in which inter-agent communication is thought of as similar to "conversations" between humans. KQML considers that each message not only contains the content but also the "intention" the sender has for that content. To see the different intentions a sender may associate with different messages, consider the example that agent *A* sends the following statement as the content of a message to agent *B*:

"The price of the book with ISBN 12-3456-78 is less than or equal to 19.95 USD"

Agent *A*, in different circumstances, may have different intentions about this statement. Agent *A* may simply

- tell *B* that this statement is true in its own database; or
- ask if this statement is true in *B*'s database; or
- ask *B* to find another agent that offers this book at that price; or
- ask *B* to *monitor* the change of the price and report back whenever this statement becomes true

KQML provides a formal specification [18, 19] for representing the intentions of messages through a set of pre-defined message types (called *performatives)* used in the messages. There are about three dozens performatives defined in the current specification of KQML. The semantics of the language have been an issue of research [20-22]. A particular agent system generally implements only a subset of these defined performatives and may introduce additional performatives of its own, as long as the new ones are defined in the same format and spirit of the KQML specification. A few common performatives are shown in Table 1.

Performative	**Meaning**
tell	The sentence in the message is true in S's knowledge base
ask-if	S wants to know if a sentence is true in R's knowledge base
advertise	S is particularly suited to processing a performative
subscribe	S wants to receive updates to R's response to a performative
recommend-one	S wants R to recommend an agent that can process the performative in the content
sorry	S cannot provide a more informed reply (i.e., S cannot answer the question asked in an earlier message from R)

Table 1. Some KQML performatives and their meaning for sender agent S and recipient R.

KQML does not impose any particular common format and interpretation on the content of the communication. In other words, the content of a message is opaque to KQML. The content language for the message body is left for the individual agents to decide. KQML allows an agent to specify in a message the language and ontology it uses for the content of a given message. Not only different agents in a system can have different content language, but different messages from the same agent can also use different content languages. Commonly used content languages include Lisp, Prolog, KIF, SQL, etc.

A KQML message is thus divided into three layers: the content layer, the message layer, and the communication layer. The content layer bears the actual content of the message, in a language

chosen by the sending agent. The communication layer encodes a set of features of the message that describe the lower level communication parameters such as the identity of the sender and recipient, and a unique identifier associated with the communication. The message layer encodes the message, including its intention (by a chosen performative), the content language used, and the ontology. The syntax of KQML messages is based on a balanced parenthesis list. The first element of this list is the performative; the remaining elements are the arguments of the performative, as keyword/value pairs. The following is an example of an actual KQML message sent by agent "joe" to agent "book-market," inquiring about the price of a particular book:

```
(ask-one
        :language       KIF
        :content        (Price 12-3456-78 ?x)
        :sender         joe
        :receiver       book-market
        :reply-with     joe-1234 )
```

In the message, the performative *ask-one* indicates that this is a query type of message. KIF is specified as the content language. The actual content is a KIF predicate, named *Price*, of two arguments: the ISBN number of the book (instantiated to 12-3456-78) and the price of the book (a variable, as indicated by the question mark preceding x). The string *joe-1234* is a unique machine generated reply id. In essence, by this message agent *joe* asks for one answer from agent *book-market* of the price of this particular book. Any reply from *book-market* to *joe* concerning this query should carry an *in-reply-to* field with value *joe-1234*. In due time, agent *book-market* might send *joe* the following KQML message if or when it has an answer to the query.

```
(tell
        :sender         book-market
        :language       KIF
        :content        (Price 12-3456-78 19.95)
        :receiver       joe
        :in-reply-to    joe-1234 )
```

The second argument of the predicate Price in the reply message is instantiated to 19.95, indicating that agent *book-market* believes (or it is true in *book-market's* knowledge base) that the price of the book identified by ISBN 12-3456-78 is 19.95. More complicated queries can be conveyed by other KQML performatives. For instance, *joe* might want *book-market* to send the information about this particular book whenever its price is updated. This can be done by a nested message, starting with the performative *subscribe*[3].

```
(subscribe
    :language       KQML
```

[3] Since this message has a KQML message as its content, its content language is specified as KQML.

```
:content      (ask-one
                    :language KIF
                    :content (Price 12-3456-78 ?x))
:sender       joe
:receiver     book-market
:reply-with   joe-2345)
```

Ontologies

Sharing the content of formally represented knowledge requires more than a shared language (such as KIF) and a communication language (such as KQML or FIPA ACL). Individual agents, as autonomous entities designed with a particular view of the world they inhabit, may have different models of the world in which objects, classes and properties of objects of the world may be conceptualized differently. For example, the same object may be named differently ("book_title" and "title" for identification of the titles of books in the databases of two different agents). The same term may have different definitions ("price" might always refer to price in USD in one agent and it might be a numeric value and currency pair in another). In addition, individual agents may conceptualize different taxonomies from different perspectives; an agent that only sells books might have a different hierarchical taxonomy than an agent that sells books, supplies and software.

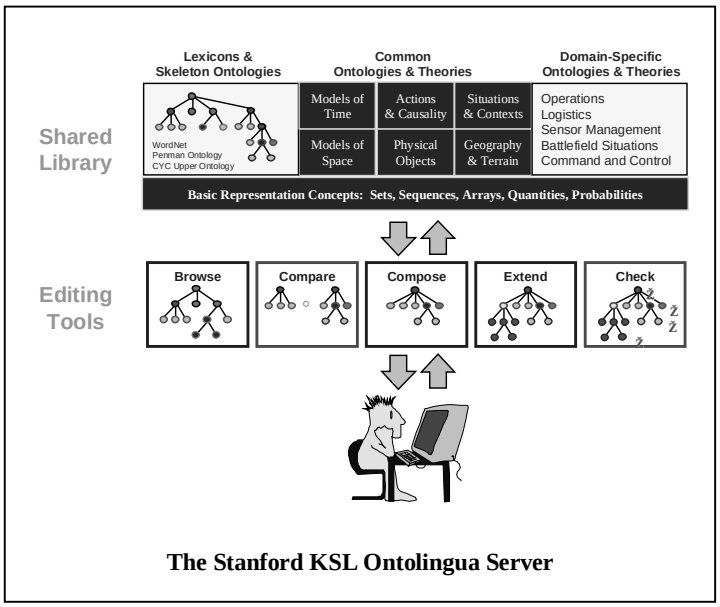


Figure 3. The Stanford KSL Ontolingua Server

Therefore, to ensure accurate mutual understanding of the exchanged messages, agents must also agree on the model of the world, at least the part of the world about which they are exchanging information with each other. In the terminology of the agent community, agents must share a common ontology. An ontology for a domain is a conceptualization of the world (objects, qualities, distinctions and relationships, etc. in that domain). Such a specification should be an objec-

tive (i.e., interpretable outside of the agents) description of the concepts and relationships that the agents use to interact with each other, with other programs, such as legacy business applications, and with humans. A shared ontology can be in the form of a document or a set of machine interpretable specifications.

As a concrete representation in a computer, an ontology consists of a specification of concepts to be used for expressing knowledge including the types and classes of entities, the kinds of attributes and properties they can have, the relationships and functions they can participate in and constraints that hold over them. Ontologies are distinguished from knowledge-bases in general not by their form, but by the role they play in representing knowledge. Ontologies are typically consensus models for a particular domain. They place an emphasis on properties that hold in all situations rather than on one or more particular ones. Consequently, their implementations put an emphasis on representing generic classes over specific instances. Ontologies do not tend to change over the course of their use in problem solving, so are well suited to compiling into programming tools and environments. Since ontologies by their nature need to support and satisfy a community of users (and developers) there is a emphasis on collaborative development and on the ability to translate them into multiple representational formalisms.

Researchers at Stanford's Knowledge Systems Laboratory have developed a set of tools and services to support the process of achieving consensus on common shared ontologies by geographically distributed groups. These tools are built around Ontolingua, a language designed for describing ontologies with it, and make use of the world-wide web to enable wide access and provide users with the ability to publish, browse, create, and edit ontologies stored on an ontology server. Users can quickly assemble a new ontology from a library of existing modules, extend the result with new definitions and constraints, check for logical consistency, and publish the result back to the library. In addition, the KSL Ontology server is able to translate Ontolingua ontologies into a number of other representation formalisms such as LOOM, CLASSIC, KIF and CLIPS. The Ontolingua Server and associated tools may be accessed at http://ontolingua.stanford.edu/.

Conclusions

Software agents offer a new paradigm for building very large scale distributed heterogeneous applications that focus on the interactions of autonomous, cooperating processes that can adapt to humans and other agents. As such, communication is key to realizing the potential of this new paradigm, just as the development of human language was critical to the development human society and culture. This holds for mobile agents as well as for stationary ones. Agents use an *Agent Communication Language* to communicate and share information and knowledge. Some [23] even go so far as to suggest that the use of a rich ACL is a defining characteristic of a soft-

ware agent – i.e., a software agent is any process that makes appropriate use an ACL to exchange information. Although this sounds circular, it is not if we stipulate that an ACL should be sufficiently rich to encode a wide range of knowledge and a reasonable set of propositional attitudes toward sentences.

Although a good deal of both theoretical and practical work on ACLs has been done in the past decade, much remains to be done. Current languages such as KQML do not address well the problem of talking about actions. KIF as a common content language does not match well with the industry's growing reliance on object-oriented programming languages and interfaces. The problem of developing common ontologies is a difficult one with many technical, social and marketplace problems yet unsolved. Adoption of software agents and their ability to communicate using high-level ACLs is inevitably a market question since agent technologies are at the focal point of company wars and they balance a demand for marketable products with a dynamic, still evolving area.

References

[1] D. S. Milojicic, M. Condict, F. Reynolds, D. Bolinger, and P. Dale, "Mobile Objects and Agents," presented at "Distributed Object Computing on the Internet" Advanced Topics Workshop, Second USENIX Conference on Object Oriented Technologies and Systems (COOTS), Toronto, Canada, 1996.

[2] C. G. Harrison, D. Chess, and A. Kershenbaum, "Mobile Agents: Are they a good idea?," T.J. Watson Research Center, Yorktown Heights, NY, Research Report 1994.

[3] J. Bradshaw, "Software Agents," : AAAI Press/The MIT Press, 1997.

[4] M. Huhns and M. Singh, "Readings in Agents," : Morgan Kaufmann, 1997.

[5] H. S. Nwana, "Software Agents: an overview," *Knowledge Engineering Review*, vol. 11, pp. 1-40, September, 1996.

[6] T. R. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition*, vol. 2, pp. 199-220, 1993.

[7] "Mobile Agent Facility (Joint Submission)," Object Management Group, Boston 1997.

[8] D. Rus, R. Gray, and D. Kotz, "Transportable Information Agents," presented at 1997 International Conference on Autonomous Agents, Marina del Ray, CA, 1996.

[9] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W. Swartout, "Enabling Technology for Knowledge Sharing," *AI Magazine*, vol. 12, pp. 36--56, 1991.

[10] R. S. Patil, R. E. Fikes, P. F. Patel-Schneider, Don McKay, Tim Finin, T. Gruber, and R. Neches, "The DARPA Knowledge Sharing Effort: Progress Report," in *Readings in Agents*, M. Huhns and M. Singh, Eds.: Morgan Kaufmann Publishers, 1997.

[11] M. Genesereth and R. F. et.Êal., *Knowledge Interchange Format, Version 3.0 Reference Manual*: Computer Science Department, Stanford University, 1992 .

[12] J. Fowler, *STEP for Data Management, Exchange and Sharing*. UK: Technology Appraisals Ltd, 1995.

[13] J. McCarthy, "First Order Theories of Individual Concepts and Propositions," in *Machine Intelligence 9*, vol. 9, D. Michie, Ed. Edinburgh: Edinburgh Press, 1979.

[14] J. Mayfield, Y. Labrou, and T. Finin, "Evaluation of KQML as an Agent Communication Language," in *Intelligent Agents II: Agent Theories, Architectures and Languages*, vol. 1037, *Lecture Notes in AI*, M. Wooldridge, J. P. Mueller, and M. Tambe, Eds.: Springer Verlag, 1996.

[15] T. Finin, Y. Labrou, and J. Mayfield, "KQML as an Agent Communication Language," in *Software Agents*, J. M. Bradshaw, Ed.: AAAI Press/ The MIT Press, 1997, pp. 291-316.

[16] J. R. Searle, *Speech Acts*. Cambridge, UK: Cambridge University Press, 1969.

[17] J. R. Searle, *Expression and Meaning: Studies in the theory of speech acts*. Cambridge, UK: Cambridge University Press, 1979.

[18] "Specification of the KQML Agent-Communication Language," ARPA Knowledge Sharing Initiative, External Interfaces Working Group July, 1993.

[19] Y. Labrou and T. Finin, "A proposal for a new KQML Specification," University of Maryland Baltimore County, Baltimore, MD Technical Report,TR-CS-97-03, 1997.

[20] Y. Labrou and T. Finin, "Semantics and Conversations for an Agent Communication Language," in *Readings in Agents*, M. Huhns and M. Singh, Eds.: Morgan Kaufmann Publishers, 1997.

[21] Y. Labrou, "Ph.D. Dissertation, Semantics for an Agent Communication Language," in *Computer Science Department*. Baltimore, MD: University of Maryland Baltimore County, 1996.

[22] I. A. Smith and P. R. Cohen, "Toward a semantics for an agent communications language based on speech-acts," in *Proceedings of the 13th National Conference on Artificial Intelligence*: AAAI/MIT Press, 1996.

[23] M. R. Genesereth and S. P. Ketchpel, “Software Agents,” *CACM*, vol. 37, pp. 48--53, July, 1994.