

DIF8914 Distributed Information Systems

Introduction of Ontology and Agent Communication

Lin Yun

Department of Computer and Information Science, Norwegian University of Science and Technology

yunl@idi.ntnu.no

Abstract

Ontologies are an emerging paradigm to support declarativity, interoperability, and intelligent services in many areas, such as Agent-based Computation, Distributed Information Systems, and Expert Systems. Agent technology is one of the most promising ways of distributing and gathering information. The reason for this is that communication and collaboration are central issues of Multi Agent System.

Agent communication languages such as ACL and KQML provide a standard for agent communication in an open MAS. These languages enable an agent to specify the intention and the content of a message as well as the protocol, the language and the ontology that are used. For the protocol and the language, some standards are available and should be known by the communicating agents.

1. Ontology

1.1 What is Ontology

Communication between different people and systems has always been a complex issue. Different needs and background contexts lead in the adoption of differing, overlapping or contradicting concepts and operation methods. This has a consequence of poor communication, limited interoperability and re-use. Therefore, there is a strong need for the reduction of conceptual confusion and the implementation of a shared understanding account [1]. Such a unifying conceptual framework would serve as the basis for communication among people and systems alike. Its role could be resembled as that of grounding to electronics: a shared reference to align with. Ontologies are designed to play that particular role.

An ontology is the term referring to the shared understanding of some domains of interest, which is often conceived as a set of classes, relations, functions, axioms and instances[2]. As such they are useful representations for knowledge on the semantic web. Ontologies can take many forms but it will include a vocabulary or terms and a definition of their meaning[3]. They are generally stored in a machine readable, textual form, but to make them easier to understand they can also be expressed as a diagram, with concepts and instances being boxes and relations being lines between them. Ontologies are primarily built to enable knowledge sharing and re-use. By committing to a common ontology, one can be sure that knowledge exchange in an area of interest can be consistent.

1.2 Ontologies as a specification mechanism

According to [4], an ontology is “a specification of a conceptualisation”. A body of formally represented knowledge is based on a *conceptualization*: the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them. A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose. Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly.

An **ontology** is an explicit specification of a conceptualization. The term is borrowed from philosophy, where an Ontology is a systematic account of Existence. For AI systems, what "exists" is that which can be represented. When the knowledge of a domain is represented in a declarative formalism, the set of objects that can be represented is called the universe of discourse. This set of objects, and the describable relationships among them, are reflected in the representational vocabulary with which a knowledge-based program represents knowledge. Thus, in the context of AI, we can describe the ontology of a program by defining a set of representational terms. In such an ontology, definitions associate the names of entities in the universe of discourse (e.g., classes, relations, functions, or other objects) with human-readable text describing what the names mean, and formal axioms that constrain the interpretation and well-formed use of these terms. Formally, an ontology is the statement of a logical theory.

We use common ontologies to describe *ontological commitments* for a set of agents so that they can communicate about a domain of discourse without necessarily operating on a globally shared theory. We say that an agent **commits** to an ontology if its observable actions are consistent with the definitions in the ontology. The idea of ontological commitments is based on the Knowledge-Level perspective. The Knowledge Level is a level of description of the knowledge of an agent that is independent of the symbol-level representation used internally by the agent. Knowledge is attributed to agents by observing their actions; an agent "knows" something if it acts *as if* it had the information and is acting rationally to achieve its goals. The "actions" of agents---including knowledge base servers and knowledge-based systems--- can be seen through a tell and ask functional interface, where a client interacts with an agent by making logical assertions (tell), and posing queries (ask).

Pragmatically, a common ontology defines the vocabulary with which queries and assertions are exchanged among agents. Ontological commitments are agreements to use the shared vocabulary in a coherent and consistent manner. The agents sharing a vocabulary need not share a knowledge base; each knows things the other does not, and an agent that commits to an ontology is not required to answer all queries that can be formulated in the shared vocabulary.

In short, a commitment to a common ontology is a guarantee of consistency, but not completeness, with respect to queries and assertions using the vocabulary defined in the ontology.

2. Agents

An *agent* is an animate entity that is capable of doing something on purpose. That definition is broad enough to include humans and other animals, the subjects of verbs that express actions, and

the computerized robots and softbots. But it depends on other words whose meanings are just as problematical: *animate*, *capable*, *doing*, and *purpose*. The task of defining those words raises questions that involve almost every other aspect of ontology.

- *Animate*. Literally, an animate entity is one that has an *anima* or soul. But *anima* is the Latin translation of Aristotle's word *psychê*, which had a much broader meaning than the English word *soul*. Aristotle defined a hierarchy ranging from a vegetative psyche for plants to a rational psyche for humans. The first question is whether Aristotle's hierarchy of psyches can accommodate the modern robots and softbots.
- *Capable*. The agent of a verb plays that role only as long as the action persists, but an entity can also be considered an agent if it has the power to perform some action whether or not it actually does. Formalizing that notion of power raises questions about modality, potentiality, dispositions, and counterfactuals that have been discussed in philosophy for centuries.
- *Doing*. The verb *do* sounds as simple as two other little verbs *be* and *have*. But like those verbs, its dictionary entry has one of the largest number of senses of any word in the English language. A common feature of all those senses is causality and purpose: some agent for some purpose causes some process to occur. This feature not only creates a cyclic dependency of *doing* on *agent*, it also introduces the notions of causality, process, and occurrence.
- *Purpose*. Purpose is defined as an intention of some agent that determines the interaction of entities in a situation. That is consistent with the definition of an agent as an entity that does something on purpose, but the circularity makes it impossible to give a closed-form definition of either term.

For the primitive terms of any theory, circular definitions are inevitable. As an example, Newton's famous equation $F=ma$ appears to define the force F in terms of the mass m and the acceleration a . Yet that same equation could be used to define the mass in terms of the force and the acceleration. Newton assumed that acceleration could be independently defined in terms of space and time, but Einstein showed that the structure of space and time itself depends on the mass of the entities in it. The fundamental concepts of any subject can only be defined implicitly by laws or axioms that express a pattern of relationships among them. Closed-form definitions are never possible for basic primitives.

2.1 Psychology of Agents

Linguistically, an agent is an animate being that can perform some action, and an action is an event that is initiated or carried out by some animate being. The circularity in those definitions can be broken by determining what characteristics of an animate being are necessary for it to play the role of an agent. Then those features can be generalized to a definition of *agent* that applies to people, animals, robots, and certain kinds of computer programs.

The word *animate* comes from the Latin *anima*, which means breath or soul. The medieval Scholastics used *anima* as a translation of the Greek *psychê*, which also means breath or soul. The basis for the modern terminology is Aristotle's treatise *Peri Psychês*, which is called *De Anima* in Latin or *On the Soul* in English. Aristotle defined the psyche as the *logos* or principle that determines what it is for something to be a living entity. Instead of a single principle of the

psyche that covered all living things, Aristotle found six related functions, which he arranged in a hierarchy: nutrition, perception, desire, locomotion, imagery, and thought:

We must inquire for each kind of living thing, what is its psyche; what is that of a plant, and what is that of a human or a beast. The reason why the functions are arranged in this order must also be considered. For without nutrition, there does not exist perception, but in plants, nutrition is found without perception. Again, without the sense of touch none of the other senses exists, but touch exists without the others, for many animals have neither vision nor hearing nor sense of smell. And of those that can perceive, some have locomotion, while others have not. Finally and most rarely, they have reason and thought. Those mortal creatures that have reason have all the rest, but not all those that have each of the others have reason; some do not even have imagery, but others live by this alone. The rational intellect requires a separate principle (*logos*). An appropriate definition of each of these functions would be the most appropriate for the psyche as well.

Aristotle's hierarchy of functions was based on his extensive study of the plants and animals known in his day. With his criteria, he was the first to recognize that sponges were primitive animals rather than plants. The subdivisions in the tree of Porphyry, are based on Aristotle's distinctions of animate/inanimate, sensitive/insensitive, and rational/irrational.

2.2 Competence Levels

Aristotle's hierarchy resembles the competence levels that Rodney Brooks (1986) defined for mobile robots. A *robot* is an AI system that receives signals from the environment and acts on the environment in a way that helps it to achieve some preestablished goals. In what he called the *subsumption architecture* for mobile robots, Brooks distinguished eight *levels of competence*, each with increasingly more sophisticated goals and means for achieving them:

1. *Avoiding*. Avoid contact with other objects, either moving or stationary.
2. *Wandering*. Wander around aimlessly without hitting things.
3. *Exploring*. Look for places in the world that seem reachable and head for them.
4. *Mapping*. Build a map of the environment and record the routes from one place to another.
5. *Noticing*. Recognize changes in the environment that require updates to the mental maps.
6. *Reasoning*. Identify objects, reason about them, and perform actions on them.
7. *Planning*. Formulate and execute plans that involve changing the environment in some desirable way.
8. *Anticipating*. Reason about the behavior of other objects, anticipate their actions, and modify plans accordingly.

Each of these levels depends on and *subsumes* the competence achieved by the earlier levels. Each level responds to signs, signals, or stimuli from the input sensors and generates output for the motor mechanisms. Yet the robot as a whole does not depend on a strict control hierarchy. The first few levels by themselves could support an insectlike intelligence that responds directly to immediate inputs without doing abstract reasoning or planning. The higher levels could inhibit the lower levels and take control for more sophisticated or intelligent behavior, but the lower levels would still be capable of automatic, reflexlike reactions to danger signals.

The behavior of the lower levels depends primarily on immediate inputs. The higher levels depend more heavily on internal representations, such as maps of the environment, memories of previous inputs, stored patterns for recognizing familiar objects, and established habits for repeatable behaviors. Every level responds to signs from the external environment and from other internal levels, but there is an increase in complexity from the automatic responses at the lower levels to the knowledge-based reasoning at the higher levels.

2.4 Artificial Psyches

Aristotle's levels may help to clarify and refine the competence levels. Nutrition, which Brooks omitted, is necessary for a robot to recharge its batteries; and desire or something like it is necessary to determine goals for the robot at every level, from the most primitive nutrition to the most sophisticated planning.

What distinguishes a software agent from an ordinary program is a unifying principle that gives it a certain autonomy. Following Aristotle, that principle may be called its *psyche*, and its definition can be based on an appropriate definition of each of its functions. The six functions of the psyche, which Aristotle applied to living things from plants and insects to humans, can serve as metaphors for the functions of artificial agents:

- *Nutrition*. For a robot or embodied agent, nutrition is the act of recharging its batteries or energy stores from time to time. For a software agent, nutrition is the procurement of computer time and storage space from a host system. A computer virus is a parasite that steals the time and space; a more benign agent lives in a symbiotic relationship with its host, providing useful services in exchange for room and board.
- *Perception*. For a robot, perception depends on input sensors and the ability to interpret the inputs. A television camera, for example, may provide a stream of data; to see, however, the robot must convert the data to a representation of objects in the environment. For a software agent, perception requires access to input devices of the host and the ability to interpret data from those devices.
- *Desire*. Aristotle's general word for desire is *orexis*, which causes an agent to reach for what is desired -- one that doesn't reach is *anorexic*. He distinguished three aspects of desire: appetite (*epithymia*), passion (*thymos*), and will (*boulêsis*). He classified appetite and passion as feelings shared with beasts and will as the result of rational thought. In their psychology of agents, Moffat and Frijda (1995) made a similar distinction between preference and will. For a software agent, the built-in equivalent of appetite or passion gives it a preference for certain kinds of states. Its will is determined by a logically derived plan for reaching a preferred state.
- *Locomotion*. For mobile robots, locomotion is a basic function that may be further divided into subfunctions, such as Brooks's competence levels. Software agents, which operate in some host system, may use the input/output devices of the host to explore the environment, including anything reachable via computer networks.
- *Imagery*. Aristotle's term *phantasia*, according to the Liddell and Scott dictionary, means the appearance, presentation, or representation of images "whether immediate or in memory, whether true or illusory." The processing of imagery by computer is an active research topic in artificial intelligence. Like Aristotle, many researchers believe that

important aspects of animal-level intelligence can be achieved by manipulating imagelike data structures rather than propositions.

- *Thought*. Aristotle reserved the highest level of the psyche for rational thought. His term for *rational animal* was *zôon logon echon* (animal having logos). For software agents, rational thought corresponds to the deductive and planning capacity that transforms the motivating forces of appetite and passion into will. A rational agent must be able to perceive relevant aspects of a situation, evaluate their desirability, and determine plans for transforming the current situation into a more desirable one.

The notion of *psyche* with its hierarchy of functions provides a framework for classifying agentive behavior. The psyche of an agent is its functional organization, and its level of sophistication depends on how much of the Aristotelian range of function it is able to support. A formal definition of the term *agent* might be based on a formalization of the informal hierarchies proposed by Aristotle and Brooks.

Computer-based systems and human operators can be seen as agents in that, while sharing physical resources, they encapsulate specialistic information (data/knowledge repositories), information processing capabilities and procedures determining their external behavior. The activities and services explicated by agents take place in a common environment, which in principle can be as large as the area covered by worldwide telecommunication networks. A realistic vision involves a number of relatively small communities of agents (including humans), connected through a network, in their turn linked to other communities to cover non internally supported needs. Starting from the seminal Minski's ideas [5], these communities of agents highly interoperating with each other can be called agencies, in that an external observer could consider the community a sort of "structured" agent offering multiple services.

While, in principle, the agent category embraces entities spanning from simple processing units (reactive agents acting on the basis of "stimulus/response" rules or mechanisms) to complex systems exhibiting "rational" behavior, we will confine ourselves to considering, as already mentioned, only the so-called cognitive agents, that is, entities endowed with symbolic representation of knowledge and driven by explicit goals. These will embody suitable procedures controlling external actions e.g. aimed at getting the resources needed to achieve perceived goals out of the environment or of other agents. For interactions like the latter, involving social ability and pro-active attitudes like commitments and intentions [6], appropriate communication capabilities become necessary, as hinted before.

Indeed, from a software engineering point of view, a software agent can be defined as an at least semi-autonomous process able to interoperate with other processes (running on the same or on a separate machine) through a suitable ACL [7; 8]. That is, we assume an intuitive general software agent model, adapted from [9] and shown in Figure 1, in which each "isolated" entity possesses one or more conveniently represented information repositories using which it can perform, without interacting with other agents, processing operations by means of suitable elaboration and control components. Information flows in an agent peculiar way along its I/O channels facing the environment (e.g. sensors and actuators).

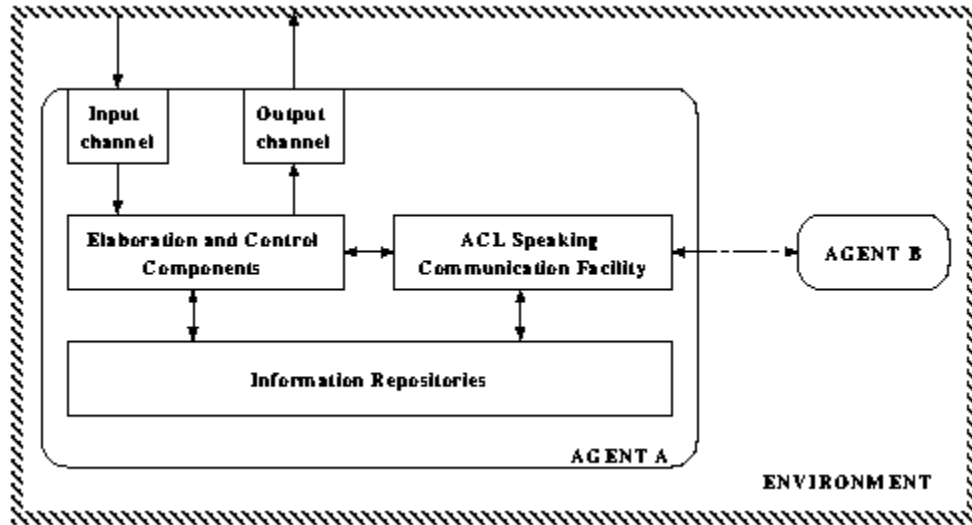


Figure 1: A general software agent model

Cooperation/coordination with other agents is achieved by extending the representation mechanisms of this isolated entity with ACL communication primitives. An essential feature of such primitives is that they must allow agent integration at the knowledge level [10], that is, in a manner independent of implementation related aspects. For the sake of simplicity, the inter-agent communication facility is shown as a self-contained module in the figure. Thus individual agents are essentially viewed as "units of knowledge and interaction" [11].

Among proposed ACLs, the Knowledge Query and Manipulation Language (KQML) developed within the Knowledge Sharing Effort [12; 13] is receiving more and more consensus both from the theoretical and from the applicative point of view. It consists of a set of communication primitives of notification (tell, untell) and of request and reply (ask- if, reply), and allows to linguistically decouple the communication modes (expressed in KQML) from the communicated information (expressed for example in KIF [14]). The content portion of a message should mention the objects and relations in a predefined conceptualization, that is in an ontology known both to the sending and to the receiving agents. These design characteristics appeared suitable for the agentification process we undertook on previously developed prototypical systems, and also for the integration of a new agent in an existing community. Therefore we strongly took inspiration from KQML while developing our experimental agent communication protocol [15].

3. Agent Communication

3.1 Why agent need communication

Realization of efficient communication between agents

When we scrutinize the case of human communication, it is very difficult for two persons to talk with each other without the knowledge of the counterpart even if they speak the same language.

If agents are deprived of the capability to communicate what they are and what they can do, agents need to go through a very inefficient process of building the models of other agents from indirect evidences.

Agent Description Ontology should make it possible for agents to communicate their attributes and capabilities. Agents can build models of other agents rapidly through the direct messaging, and then efficient communication between them can be established.

Realization of agent services

In order to realize the agent services such as brokerage/recruitment/recommendation, agents need to have accurate models of other agents.

KQML specification [16] defines advertise messages. KQML specifies that an advertise message has the KQML message as its content. Therefore an advertise message can only communicate the message-handling capability of the sender agent by a matching pattern. If we restrict the agents to the advertise messages as the ways to convey their capabilities, this would greatly restrict the possibilities of agent services. It should be very difficult to realize content-based brokerage. We need predicates, of which semantics are defined in the ontology, to provide a variety of such agent services.

Human benefits

It would benefit the people, especially the developers and users of agent systems in the current situation where the definitions of agents are just abundant.

3.2 Agent Description Ontology

Name of the ontology

We need to give the name of the ontology that should appear as the value for the :ontology parameter of the ACL message. If the terms are appropriately categorized, we need to give names for ontologies corresponding to those categories.

Framework to express agent models

We need the framework to express agent models, which the ontology will be based on. This may be a meta-ontology in one of first order logic languages. In order to make the standard language-independent, we can take such an approach for example to provide models in graphs and serializations of the models into SL, KIF, XML [11], etc. This is a similar approach to that of RDF (Resource Description Framework) [10].

Content of the ontology

We need to standardize the content of the ontology, which is the essential part of this proposal. We need the terms for the predicates to describe the agents and possibly the values or the formats of the values for those predicates. Here we list the categorized list for the possible terms. These categories may not be exhaustive and may be reorganized during the discussion.

Terms for describing agent's knowledge

– Range of agent's VKB (Virtual Knowledge Base)

- * Syntactical range of VKB
- * Semantic range of VKB

Terms for describing agent's capability

- Inference capability
- Protocols the agent can handle
- Communicative acts the agent can handle
- Languages the agent can handle
- Mobility related capability
- Access methods (email address, URL)

Terms for describing agent's attributes

- Groups the agent belongs to
- Specialty of the agent
- History of the agent
- Access restrictions of the agent

3.3 Types of agent communities

In [17; 18], there are 3 different agent society types. These are used as basis for architecture definition of different MASs. The following paragraphs illustrate the generic characteristics of these organizations.

Hierarchical

This type of organization separates agents in two distinct categories: *representatives* and *resource-dedicated*.

Representative agents act as the mediators between the rest of the MAS and the resource-dedicated agents they are in charge of breaking up complex queries for distribution and compiling results into replies. Resource-dedicated agents access a local information resource, like an ontology or DB, and reply to representative requests.

In this scheme information is distributed among the resource-dedicated agents, as each one of them accesses a particular source of information. This approach allows modularity in information storage and easier maintenance. Workload is distributed among the resource-dedicated agents, while information combination is left to the representatives.

Representatives do not need to hold a repository of information. This approach, although managing workload distribution more efficiently, adds a lot of network overhead.

Peer-to-peer

In this organization scheme, all participating agents establish peer-to-peer relations among them. There is no distribution of roles among agents and so outsiders can contact any agent they wish, in order to retrieve information. This means that all of the community's agents have to be known to the outside in order to be accessed. This is a big difference from the previous architecture, where only representative agents were required to be accessible from outside the community; the rest of the agents remained opaque to the outer world. Each agent co-operates with the others in the community in order to respond to complex queries that cannot be handled by one agent alone. Formally, there is only one type of agent. The only distinction that can be made is through the type of the underlying resource that each agent accesses.

This is another difference with the previous typology, since each agent can be considered as a combination of both a representative and a resource-dedicated agent.

Replication

In this type of agent organization, which is a more restrictive version of the previous, all agents keep a local copy of all available information resources. Therefore, there is only one type of agent, both technically and virtually, since there can be no distinction between agents, neither in terms of functionality nor depending on the type underlying resources it has access to. This implies that each agent is as self-contained as can be, not requiring to contact another agent to resolve a query. Content is replicated across the system, resulting in high information redundancy. This puts a high demand on resources on the one hand, but ensures better fault tolerance than the other two schemes, while network usage is kept to the absolute minimum. It is obvious that this scheme cannot be applied to even medium-sized knowledge domains, since, in most cases, information redundancy is something intolerable.

4 Agent Communication Languages

To be useful, agents need to be able to interact with their surroundings, be it human users, system services or other agents. In order to communicate we need language. Mayfield, Labrou and Finin have postulated a number of desiderata for a language that an agent might use for interactions with its environment [19]. In the paper the following general categories are identified (all explanations are quotes from the paper):

Form	<i>A good agent communication language should be declarative, syntactically simple and readable by people.</i>
Content	<i>A communication language should be layered in a way that fits well with other systems. In particular, a distinction should be made between the communication language, which expresses communicative acts, and the content language, which expresses facts about the domain.</i>
Semantics	<i>Semantics is an issue that has often been neglected during the design of communication languages. [...] The semantics of a communication language should exhibit those properties expected of the semantics of any other language. It should be grounded in theory, and it should be unambiguous.</i>
Implementation	<i>The implementation should be efficient, both for speed, and for bandwidth utilization</i>
Networking	<i>An agent communication language should fit well with modern networking technology.</i>
Environment	<i>The environment in which intelligent agents will be required to work will be highly distributed, heterogeneous, and extremely dynamic.</i>
Reliability	<i>A communication language must support reliable and secure communications among agents</i>

These desiderata give a hint as to the complex issues involved in getting agents to successfully communicate. Another important language issue for agents is the *implementation language*.

In the following we will approach the issue of communication using a fairly general approach. Figure 2 shows a model for communication where an originator **A** sends a message **M** to a recipient **B**.

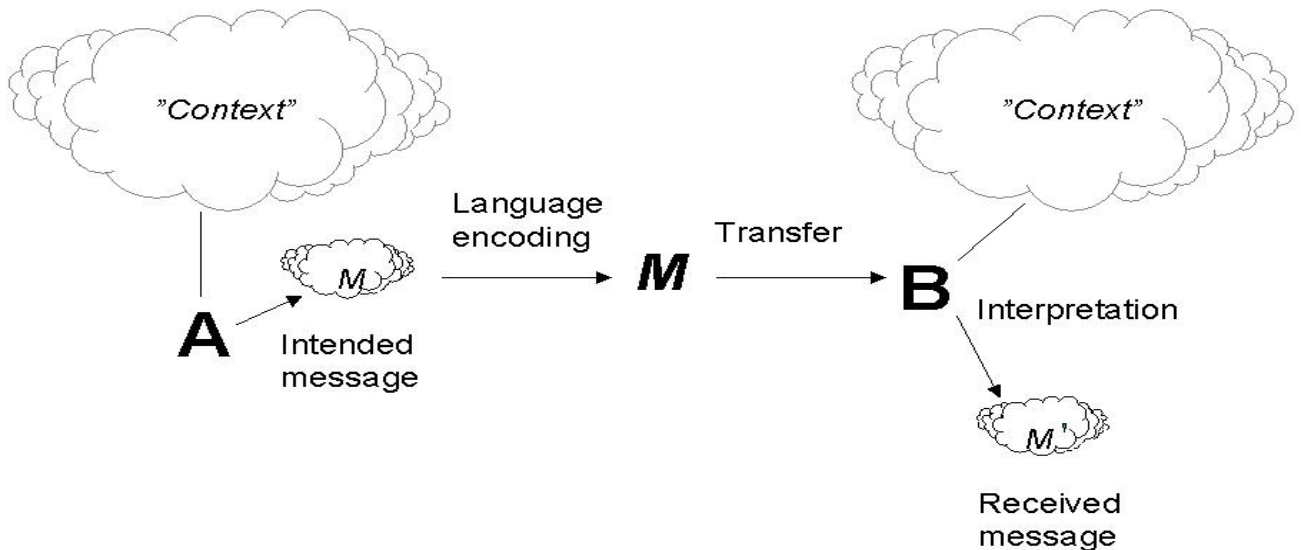


Figure 2. General communication model

For simple applications the messages communicated to and from agents may be simple. However, as agent applications get more complex, the need arises to express and maintain more complicated beliefs and intentions. For instance, an agent designed to react autonomously on events in its environment need to maintain some model of its environment as well as a set of rules governing its behavior. One way to achieve this is to use a **knowledge representation** language. Another important use of knowledge representation languages is for exchanging messages between agents.

As for any communication an important obstacle to overcome in order for agents to communicate, is to establish a common frame of reference, what is labeled "*Context*" in figure 2. It is the context that determine how a message is interpreted. To address this problem, we try to conceptualize the world using **ontologies**.

For successful interoperation the means and methods for the actual exchange of a message needs to be standardised. For this purpose we make use of a **communication protocol**.

3.1 Knowledge representation

To maintain and use knowledge, beliefs and intents, an agent needs a way of expressing them. For this purpose we need a *knowledge representation language*. These languages come in a number of flavors, here we will mention two important representatives, *Descriptions Logics* and the *Knowledge Interchange Format*.

Description Logics (DL) is a powerful formalism for expressing concepts and their interrelationships [20]. In DL, concepts are organized into IS-A hierarchies. Concepts are specifications such that given an individual (object instance) a DL system can *recognize* the individual and determine which concepts it belongs to. DL systems also perform subsumption checking.

The *Knowledge Sharing Effort* (KSE) is "... a consortium to develop conventions facilitating sharing and reuse of knowledge bases and knowledge-based systems. " [21]. Interest areas for KSE include:

- Common Knowledge Base (KB) languages
- Common constructs among languages
- Shared and reusable KBs
- External interfaces

An important effort with KSE has been the development of the *Knowledge Interchange Format* (KIF) [22; 23]. KIF is based on first order predicate calculus and has a LISP-like prefix syntax. KIF is capable of expressing simple facts as well as more complex relationships. The language includes a number of logical operators that make it possible to express rules of various kinds. Furthermore, KIF provides constructs for describing procedures, i.e. programs to (possibly) be executed by an agent. Below are a number of KIF examples from [23].

Facts

```
(salary 015-46-3946 widgets 72000)
(salary 026-40-9152 grommets 36000)
(salary 415-32-4707 fidgets 42000)
```

These are assertions about the employees Widgets, Grommets and Fidgets respectively. For instance, Mr. Widgets has employee number 015-46-3946 and has a salary of \$72.000 per annum.

An asserted relation

```
(> (* (width chip1) (length chip1))
    (* (width chip2) (length chip2)))
```

This relation states that the area of chip1 is greater than the area of chip2.

Rule

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

This is a rule that allows a system to deduce the sign of x raised to n if x is real number and n is even.

Procedure

```
(progn (fresh-line t)
      (print "Hello!")
      (fresh-line t))
```

This is a piece of code that will cause the a newline, the phrase **Hello!** and another newline to be output to the standard output.

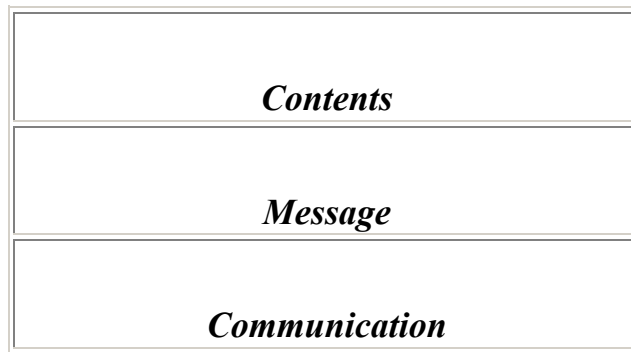
KIF is a popular language. For instance, systems and parsers are available for Lisp and Java [22].

3.2 Communication protocols

Once we have achieved a way of representing the knowledge of our agents, we need tools for sharing and exchanging that knowledge. There are two main initiatives to this end.

The *Foundation for Intelligent Physical Agents* (FIPA [24]) has a proposed an agent communication language (ACL) that is founded on Speech Act Theory. The FIPA-ACL abstracts away low level communication details and assumes the existence of an Agent Management System not part of the languages.

The *Knowledge Query and Manipulation Language* (KQML) is perhaps the most widely used communication formalism [25; 26]. KQML was developed as a part of the Knowledge Sharing Effort [21]. KQML is built around a number of *performatives* designed to achieve tasks at three conceptual layers:



The attempt in KQML to be able to handle tasks at several levels of abstractions is sometimes put forward as a significant drawback of the language. Distinction between layers are not directly evident in programs and specifications, something that is confusing and make building good abstractions difficult. In [27] Mayfield, Labrou and Finin evaluate KQML against a set of desiderata for agent communication languages proposed by the authors.

The performatives, or KQML instructions, can be divided into several categories. The most important categories are:

- Queries** These performatives are used to send questions for evaluation somewhere.
- Responses** Used by a agent in order reply to queries and request.

Informational Informational performatives are used to transfer information.

Generative This class of performatives is used for controlling and initiating the exchange of messages.

Capability definition Allows an agent to learn about the capabilities of other agents and to announce it own to the agent community.

Networking Networking performatives make it possible to pass directives to underlying communication layers.

As an example, in the following KQML expression, Joe queries a *stock-server* for the current price of an IBM share.

```
(ask-one :sender joe :content (PRICE IBM ?price)
        :receiver stock-server
        :reply-with ibm-stock :language LPROLOG
        :ontology NYSE-TICKS)
```

In response to this query, the stock-server might use the following expression to tell Joe that the current price is "14":

```
(tell :sender stock-server
      :content (PRICE IBM 14)
      :receiver joe
      :in-reply-to ibm-stock
      :language standard_prolog
      :ontology NYSE-TICKS)
```

On the whole, the KQML language provides a fairly low level foundation for the design of complex agents and agent framework. A number of such implementations exists, many of which are freely available [28].

4. Reference

1. Gandon F., *Ontology Engineering: A Survey and a Return on Experience*, Rapport de Recherche, INRIA, March 200
2. T.R. Gruber (1993) "A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5 199-220
3. Mike Usehold and Michael Gruninger (1996) "Ontologies: Principles. Methods and Applications", *Knowledge Engineering review*. 11, 2
4. Gruber, T.R. "A Translation Approach to Portable Ontology Specifications". *Knowledge Acquisition*, 5(2):199-220, 1993.
5. Minski, M. (1985). *The Society of Mind*, Simon and Schuster, New York.
6. Woolridge, M., and Jennings, N. R. (1995). *Intelligent Agents: Theory and Practice*. *The Knowledge Engineering Review*, 10(2):115-152.
7. Genesereth, M. R. (1992). *An Agent-Based Approach to Software Interoperability*. In *Proceedings of the DARPA Software Technology Conference*, pages 359- 366, Meridian Corporation, Arlington, VA.
8. Genesereth, M. R., and Ketchpel, S. P. (1994). *Software Agents*. *Communications of the ACM*, 7(37):48-53.
9. Burmeister, B., and Sundermeyer, K. (1992). *Cooperative Problem-Solving Guided by Intentions and Perception*. In Werner, E. and Demazeau, Y. (Eds.), *Decentralized A.I.* -3,

- Proceedings of the Third European Workshop on Modelling Autonomous Agents in a Multi-Agent World, pages 77-92, Elsevier, Amsterdam.
10. Newell, A. (1982). The Knowledge Level. *Artificial Intelligence*, 18:87-127.
 11. Gasser, L. (1991). Social Conceptions of Knowledge and Action: DAI Foundations and Open Systems Semantics. *Artificial Intelligence*, 47:107-138.
 12. Finin, T., McKay, D. and Fritzson, R. (Eds.) (1992). The KQML Advisory Group. An Overview of KQML: A Knowledge Query and Manipulation Language. Draft.
 13. Finin, T., Weber, J., Wiederhold, G., Genesereth, M. R., Fritzson, R., McKay, D., McGuire, J., Pelavin, P., Shapiro, S. and Beck, C. (1993). Specification of the KQML Agent Communication Language. Technical Report EIT 92-04, Enterprise Integration Technologies, Palo Alto, CA.
 14. Genesereth, M. R., and Fikes, R. E. (1992). Knowledge Interchange Format, Version 3.0 Reference Manual. Technical Report Logic 92-1, Computer Science Department, Stanford University.
 15. Lanzola, G., Falasconi, S., and Stefanelli, M. (1995). Cooperative Software Agents for Patient Management. In Barahona, P., Stefanelli, M., and Wyatt, J. (Eds.), *Proceedings of the 5th Conference on Artificial Intelligence in Medicine Europe (AIME95)*, Lecture Notes in Artificial Intelligence, vol. 934, pages 173-184, Springer-Verlag.
 16. Tim Finin and et al., The DARPA Knowledge Sharing Initiative External Interfaces Working Group, "Specification of the KQML Agent Communication Language," 1994/2/9,
 17. Gandon, F. et al, A Multi-Agent System to Support Exploiting an XML-based Corporate Memory, Proc. Of the 3rd Int. Conf. on Practical Aspects of Knowledge Management, Basel, Switzerland, 30-31 Oct. 2000
 18. Klush, M., *Intelligent Information Agent: Agent-based Information Discovery and Management on the Internet*, Springer, 1999
 19. James Mayfield, Yannis Labrou, and Tim Finin. *Desiderata for Agent Communication Languages*. Proceedings of the AAAI Symposium on Information Gathering from Heterogeneous, Distributed Environments, AAAI-95 Spring Symposium, Stanford University, Stanford, CA. March 27-29, 1995.
 20. Patrick Lambrix, *Description Logics*, WWW-page: <http://www.ida.liu.se/labs/iislab/people/patla/DL/index.html>
 21. Robert Neches, *The Knowledge Sharing Effort by Robert Neches* , WWW-page: <http://www-ksl.stanford.edu/knowledge-sharing/papers/kse-overview.html>
 22. UMBC AgentWeb, *KIF, Knowledge Interchange Format*, WWW-page: <http://www.cs.umbc.edu/kse/kif/>
 23. Finin, Labrou and Mayfield, *KIF101 -- A brief introduction to the knowledge interchange format* , WWW-page: <http://www.cs.umbc.edu/kse/kif/kif101.shtml>
 24. Foundation for Intelligent Physical Agents, WWW-home page: <http://drogo.cse.stet.it/fipa/>
 25. UMBC KQML Web, WWW-page: <http://www.cs.umbc.edu/kqml/>
 26. Tim Finin, Rich Fritzson, Don McKay, and Robin McEntire. *KQML - A Language and Protocol for Knowledge and Informational Exchange*. Technical Report CS-94-02, Computer Science Department, University of Maryland. <http://www.cs.umbc.edu/kqml/papers/kbkshtml/kbks.html>
 27. James Mayfield, Yannis Labrou, and Tim Finin. *Desiderata for Agent Communication Languages*. Proceedings of the AAAI Symposium on Information Gathering from

Heterogeneous, Distributed Environments, AAAI-95 Spring Symposium, Stanford University, Stanford, CA. March 27-29, 1995.

<http://www.cs.umbc.edu/kqml/papers/desiderata-acl/root.html> or

<http://www.cs.umbc.edu/kqml/papers/desiderata-acl.ps>.

28. KQML Software, WWW-page: <http://www.cs.umbc.edu/kqml/software/>