Agents on the Web

Conversational Agents

Michael N. Huhns and Munindar P. Singh

March 1, 1997

When you build or buy an agent for the web, you want it to perform as well as possible. Increasingly, this means your agent should take advantage not only of all the information resources in the web, but also of all the other agents that might be operating there.

And, soon there will be a lot of them. The computational architecture that seems to be evolving out of an informationally chaotic web consists of numerous agents representing users, services, and data resources. A typical paradigm of usage is as follows: resource agents advertise to the services; user agents use the services to find the resource agents, and then query them for the information they need.

Agents representing different users might *collaborate* in finding and fusing information, but they might *compete* for goods and resources. Similarly, service agents may collaborate or compete with user, resource, and other service agents.

Whether they are collaborators or competitors, the agents will be interacting purposefully with each other. Most purposeful interactions—whether to inform, query, or deceive—require that the agents talk to one another, and talking intelligibly requires a mutually understood language.

"Lingua Franca"

Agent projects have investigated communication languages for a number of years. Early on, agents were local to each project, and their languages were mostly idiosyncratic. Now, the challenge is to have your agent talk to anyone's agents, not just to your own. The obvious solution is a lingua franca—ideally, all the agents who implement the (same) lingua franca will be mutually intelligible.

What should such a lingua franca be like? It should have an unambiguous syntax or form, so the agents can all parse sentences the same way. It should have a well-defined semantics or meaning, so the agents can all understand sentences the same way. It should be well known, so different designers can implement it, and an agent has a chance of encountering another one who knows it. And, it should have the expressiveness to capture the kinds of things agents may need to say to one another.

As you can see, this is a nontrivial list of requirements. Of these, the easiest is coming up with an unambiguous syntax. Being well known is not so much a technical as a political

requirement on a language—committees and consortia are expected to ensure that their results will be widely adopted. Ensuring the expressive power of a language is potentially very difficult, but it turns out that we can borrow a lot of good ideas from the study of human language. That leaves the question of meaning—no one has quite figured that out, and we will soon explain why.

So what language should you give your agent (or teach it—the subject of a future column), so that it will understand and be understood? At the moment, there are two main choices: KQML and the FIPA specification. Both are attempts to separate the domain-dependent part of a communication—the content—from the domain-independent part—the packaging—and then to provide a standard for the domain-independent part. Both are based on "speech acts"—more on these below. Neither has been completed, although KQML was begun earlier and is more mature. Both are being formalized, so that independently developed agents conforming to their standard will understand each other. Both have prototype implementations, initial applications, and their own adherents.

BEGIN SIDEBAR

"Speech Acts"

Speech acts have to do with communications—they have nothing to do with speech as such, except that human communication often involves speech. Speech act theory was invented in the fifties and sixties to help understand human language. The idea was that with language you not only make statements, but also *perform actions* <Austin62>. For example, when you request something you don't just report on a request, but you actually cause the request; when a justice of the peace declares a couple man and wife, she is not reporting on their marital status, but changing it. The stylized syntactic form for speech acts that begins "I hereby request ..." or "I hereby declare ..." is called a performative. With a performative, literally, saying it makes it so! The interesting thing is that verbs that cannot be put in this form are not speech acts. For example, "solve" is not a performative, because "I hereby solve this problem" just doesn't work out—or Math students would be a much happier lot!

Several hundred verbs in English correspond to performatives. This obviously calls for classifications, and lots have been given. For most computing purposes, speech acts are classified into assertives (informing), directives (requesting or querying), commissives (promising), prohibitives, declaratives (causing events in themselves, e.g., what the justice of the peace does in a marriage ceremony), expressives (expressing emotions).

In natural language, it is not easy to determine what speech act is being performed. For example, if Mike says "It's cold here," he might be telling you about the temperature, or he may be requesting you to turn up the heat. This is one of the reasons why natural language is tricky. In artificial languages, we do not have this problem.

<Austin62> John L. Austin, *How to Do Things with Words*, Oxford: Clarendon, 1962.

END SIDEBAR

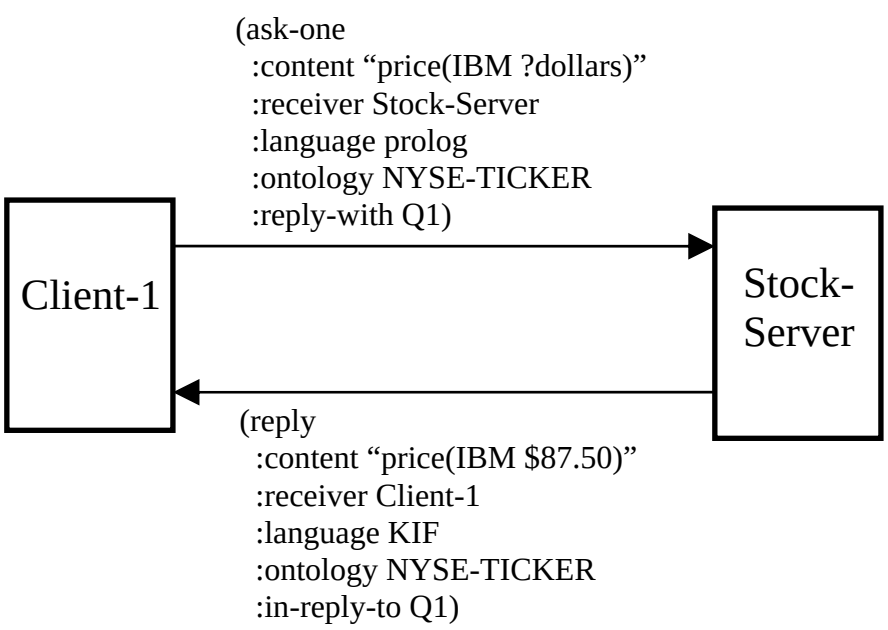
"KQML"

The Knowledge Query and Manipulation Language, KQML, <KQML1> was defined under the DARPA-sponsored Knowledge Sharing Effort. KQML assumes a layered architecture. It assumes, at the bottom, functionality for message transport or communication. It leaves, at the top, the content to be specified by the applications, typically in some formal language, such as the Knowledge Interchange Format (KIF) <KIF> or Structured Query Language (SQL) <SQL> for databases. It provides, in the middle, the primitives with which agents can exchange meaningful message. In other words, KQML provides a way to structure the messages, but lets the agent designers worry about what is in them.

KQML provides 31--41 "performatives" <KQML1, KQML2> in a slight misuse of terminology (see the sidebar on speech acts). Although varied, KQML's performatives are all assertives and directives. They fall into a few major classes. One class, which includes **`tell`**, **`evaluate`**, and **`subscribe`**, is geared toward the actual communication of content. Another class includes primitives to control the flow of information by, for example, sending an explicit **`next`** each time another answer is wanted from a source agent. A third class allows for **`recruit`**_ing agents and performing other brokering and facilitating functions.

KQML assumes the message transport is reliable and preserves the order of messages, but does not guarantee delivery times. For this reason, the underlying paradigm of communication is asynchronous; at the application-level, the effect of synchronous communication is achieved by tagging messages to relate them, e.g., tagging responses to corresponding queries. In this way, KQML supports some elementary interaction protocols, although more sophisticated protocols must be built external to KQML.

The KQML semantics is given informally, although formalizations are underway. KQML agents are assumed to have a virtual knowledge base (VKB) containing beliefs and goals. They can communicate about the virtual knowledge base of themselves and others. Thus, a **`tell`** reports on the contents of its VKB; and an **`evaluate`** directs the recipient to report on its VKB.



An example of a message exchange in KQML.

"FIPA"

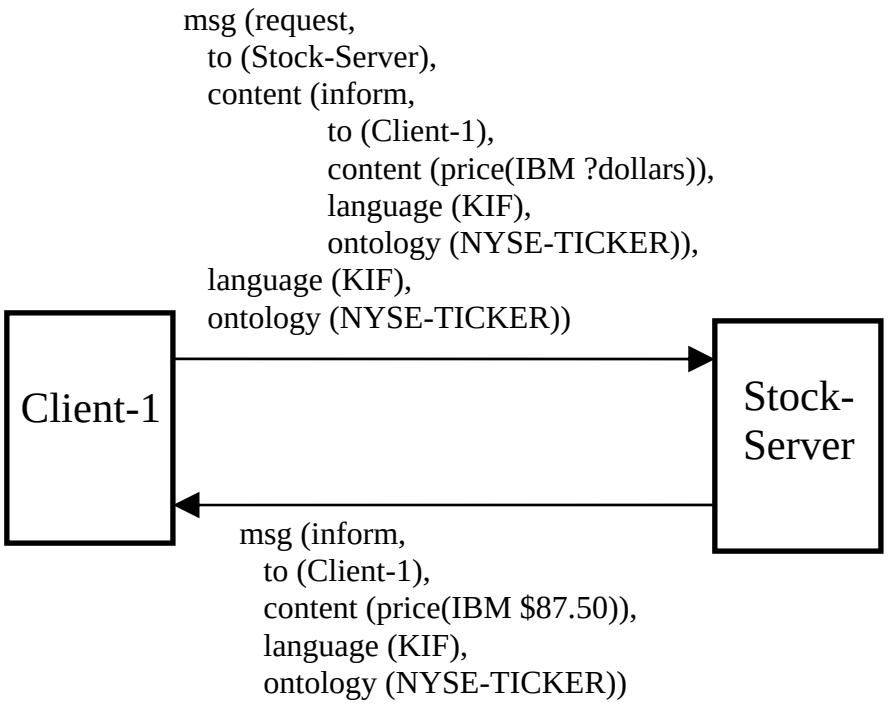
An alternative speech-act-based language was proposed by the Foundation for Intelligent Physical Agents <FIPA>. FIPA is a consortium with a number of European and Asian, and some American members.

FIPA specifies a smaller set of performatives than KQML—6—but they can be composed to enable agents to express more complex beliefs and expectations. For example, an agent can **`request`** to be **`inform`**_ed about one of several alternatives. The performatives deal explicitly with actions, so requests can be for communicative actions to be done by the message recipient.

The FIPA specification comes with a formal semantics. This, in general, is a strong point. For example, it guarantees that there is only one way for an agent to achieve any of its communication goals. Without this guarantee, agents (and their designers) would have to choose among several alternatives, leading to potential misunderstandings and unnecessary work.

However, the FIPA semantics is specific to a certain kind of agent, behaving in a certain manner. The agents must be sincere and they must not make assertions unless they are absolutely certain about their belief. For example, FIPA restricts the agents as to when they can **`inform`** another agent (when they know something and the other party does not). This would eliminate broadcasts, because an agent would not know that every listener did not know

or was uncertain about its assertion. Moreover, an engineer agent could not discuss design possibilities, while a politician agent might only make assertions when he is sure his audience already believes what he is saying! This overly legislates the society in which the agents live. We know which is the right behavior for our agents—it is not the function of the protocol to determine the behavior.



An example of a message exchange using the FIPA specification.

"Evaluation"

So, which should you choose? KQML suffers from, as yet, poorly defined semantics. As a result, of the many implementations of KQML, each seems unique. This makes communication difficult, and your KQML agent might not be understood. Also, security has not been addressed in KQML: there are no provisions to authenticate agents or guarantee the integrity of KQML messages.

The FIPA specification, by contrast, attempts to formalize the semantics and provides a security model. However, in view of its recency, it has not been widely tested or adopted. As a result, your FIPA agent might not find anyone to communicate with.

The essential semantic difference is that with the FIPA specification, *Agent A tells Agent B something, because A wants B to believe it*; whereas, with KQML, *Agent A tells Agent B something, because A wants B to know A believes it*. KQML is, in this sense, more cautious.

We suggest, if you don't have the option to wait for a consensus to emerge, you choose KQML due to its current lead in market share, and then hope for continued effort in standardizing its semantics. And when dealing with agents designed by others, be sure to use the same dialect!

“System of the Bimonth”

There is a shell available for download that enables you to construct Java applets that can converse in KQML <JAT>. Check it out!

"References"

<FIPA> FIPA home page: <http://drogo.cselt.stet.it/fipa/>

<JAT> Java Agent Template: <http://cdr.stanford.edu/ABE/JavaAgent.html>

<KIF> Knowledge Interchange Format: <http://hpdce.stanford.edu/newkif.html>

<KQML1> KQML home page: <http://www.cs.umbc.edu/kqml/kqmlspec/spec.html>

<KQML2> James Mayfield, Yannis Labrou, and Tim Finin, “Evaluation of KQML as an Agent Communication Language,” in J.P. Muller and M. Tambe (eds.), *Intelligent Agents II: Agent Theories, Architectures, and Languages*, Springer-Verlag, Berlin, 1996.

<KSE> Ramesh S. Patil, et al., "The DARPA Knowledge Sharing Effort: Progress Report," in *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, 1992.

<SQL> Ramez Elmasri and Shamkant Navathe, *Fundamental of Database Systems*, 2nd edition, Benjamin Cummings, Redwood City, 1994.