KQML Lite Language and Services

Robin McEntire and Donald P. McKay

Advanced Software Technology and Research
Lockheed Martin
590 Lancaster Ave
Frazer, PA 19355-1808
Robin.A.McEntire@lmco.com and Donal.P.McKay@lmco.com

Introduction

Research on the Knowledge Query and Manipulation Language (KQML) is one of the longest on-going efforts in agent communication language standardization. Recently, the KQML community and the international organization Foundation for Intelligent Physical Agents (FIPA) have begun to coalesce lessons learned from the initial agent communication language endeavors. One aspect of this work is the integration of KQML and FIPA agent communication languages that address many of the perceived shortcomings of KQML and the complexity of the FIPA framework. We have developed an initial specification for KQML Lite – a near-term practical effort to provide a practical, reasonable, sharable and useful agent communication language specification and tool set. KQML Lite is not intended to be the final answer on agent communication languages, but rather it attempts to identify one place where commonality could prove useful.

The major issues we have addressed include:

• The distinction between communicaiton performatives, i.e., speech acts, and the communication actions elaborated within KQML.

• The variance about what the individual KQML primitives mean, and, how they might be used coherently in series to support agent interactions.

• A precise and unambiguous specification promoting more widespread use and adoption.

• the potential unification of KQML dialects such as “KQML Classic” and “KQML ‘93”. The basic issue revolves around two items; is there some outer “message layer” which encapsulates some inner communication actions layer and is the language specification clear and unambiguous when one embeds or composes expressions in the language.

Identify and leverage tools and current systems so that the agent community can rapidly evolve a new set of tools supporting a next round of research and development in agent communication languages, i.e., KQML right of FIPA Agent Communication Language.

We have developed an initial KQML Lite specification and describe its relationship to FIPA Agent Communication Language.

KQML Lite

An agent communication language, such as KQML, provides a set of generic communication primitives and their associated protocols for use by any agent. However, a system of agents also needs a set of services that define both necessary infrastructure components that allow agents to interoperate properly and effectively, and other agents that have become accepted components in many agent-based systems. We define a number of such agent services in terms of the KQML Lite language primitives and a specific set of communication primitives associated with each service that must be supplied by that service.

KQML Lite Services

[put something here to introduce the services]

Agent Naming Service (ANS)

The ANS provides agent registration services for agents and provides mappings of agent names to addresses for as many transport mechanisms as are supported by each agent.

Proxy Service

A Proxy agent provides message routing capabilities to agents that are otherwise unable to provide their own routing functionality. The Proxy agent would be used, for example, with agents that are implemented as Java applets, since an applet has, by definition, highly restricted routing ability. In addition to routing both incoming and outgoing messages for such “agentlets”, a Proxy agent has the ability to register an agentlet with a local ANS, if an agent requests such. The primary duty of the Proxy Agent is to route messages, both from agentlets for which it is providing the routing functionality to the proper receiving agent and also from full-up agents to the appropriate applet.

Broker Agent

The service offered by a Broker Agent is to determine an appropriate receiving agent or agents, for a given message. A Broker will find a match by associating the content expression of the brokering request with attributes of an agent that it has advertised to the agent community. A

Broker may support one or more of the communication primitives from the set of Brokering primitives, that define behavior such as *recommend*ing, *recruit*ing and *broker*ing.

Management and Control Service

An Agent Manager/Controller has the ability to start, stop, suspend and resume an agent or a system of agents. An intelligent version of this agent could detect when agents have ceased to respond to messages or has died, and, if appropriate, restart an agent.

KB/DB Wrapper Service

A KB/DB Wrapper Agent services requests for access to a data or knowledge store for the purpose of querying or modifying that data or knowledge store.

Logging Service

A Logging Agent acts as a recipient for all incoming and/or outgoing message traffice for an agent or agents. Therefore, the content of a message sent to the logging agent would be the complete KQML message that had been received by, or sent out by, an agent. The Logging Agent does not care about the semantics of the message that it receives to be logged, and is only responsible for making the logged message persistent.

Timings Service

A Timings Agent acts as a recipient of *timings* messages from other agents. A *timings* message carries a description of how much time was spent in defined sections of an agent for processing a given message. Minimally, four times are captured, namely;

1. Time spent reading an incoming message
2. Time spent processing an incoming message
3. Time spent writing an outgoing message
4. Time spent waiting for a response to a message (for messages which have requested a reply)

Other times may be included in the timings message, which comprises the content-level expression of the message sent to the Timings Agent. These additional times may capture application-level or API-level processing times.

Additional Agent Services

We consider the set of agent services described previously to be an initial set that should be expanded as needs arise and the experience of building agent-based systems grows. Candidates for future additions to the current agent services include:

- Conversation/Protocol Service
- Security Services
- Mobile Agent Docks
- Gateway Services (e.g., services that allow the interoperation among agents using different agent communication languages and/or protocols)
- Agent Debugging Services
- Various Mediation Services

Conclusions

We have developed a KQML agent tool kit based on a prior specification of KQML, which encompasses these services and features. It has interoperable components and interfaces in Java, C, C++, Common Lisp, and Perl. This implementation has been licensed to nearly three dozen research labs and universities worldwide. We expect to have developed at least a Java implementation and Java accessible services for KQML Lite by the time of the Tools Workshop.

Acknowledgments. This work was funded under contract to the Defense Advanced Research Projects Agency (DARPA).