2011 International Conference on Innovation, Management and Service
IPEDR vol.14(2011) © (2011) IACSIT Press, Singapore

A Survey on Agent Communication Languages

Sandip Vaniya[1], Bhavesh Lad[2] and Shreyansh Bhavsar[2]

[1]Deptt. of Information Technology, A.D. Patel Institute of Technology, V.V. Nagar- 388120, India
[2]Deptt. of Computer Engineering, A.D. Patel Institute of Technology, V.V. Nagar- 388120, India

Abstract. In this paper we address the issue of agent communication languages used in multi-agent systems and The Knowledge Query Manipulation Language (KQML) in particular .Based on speech Act theory , we present idea of agent communication language and various requirements which enhance the quality of ACL. How KQML packages the original content of a message without bothering about the syntax or format of the content message, is described. Overlapping the content by KQML syntax and then according to sender and receiver information it will be sent to relative receiver. We discussed different performatives of this language which are classified into main seven categories. This paper includes Foundation for intelligent physical agents(FIPA) and its own language FIPA ACL which is very similar to KQML. As KQML is widely used in agent systems, various applications of this language are also included.

Keywords: speech act theory, performatives, communication facilitators .

1. Introduction

Multi-agent systems (MAS) are the subject of research for researchers studying system made up of multiple heterogeneous intelligent entities called “Agents”. The agent in MAS can compete, cooperate or simply coexist. MAS differs from distributed problem solving in the sense that there is no common goal to be solve which is known at design time; on a contrary, a multi-agent system is generally populated by different agent having different purposes.MAS application range from information throw industrial process control to electronic commerce all this applications have one thing in common agent must be able to “talk” to each other to decide what action to take and how this action can be coordinated with others’ actions. The language used by the agents for this exchange is the agent communication language (ACL). An ACL stems from the need to coordinate the action of an agent with that of the other agents. It can be used to share information and knowledge among agent in distributed computing environment, but also to request the performance of the task. The main objective of ACL is to model a suitable framework that allows heterogeneous agents to interact, to communicate with meaningful statements that convey information about their environment or knowledge.

[1]Sandip Vaniya. Tel.: + 91 7567595942; fax: +91-2692 -238180
E-mail address: s.vaniya@gmail.com.

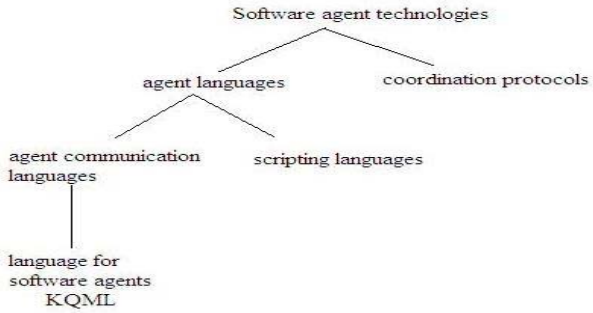

Fig. 1: A taxonomy of agent related technologies.[1]

2. Overview

When we describe agents as intelligent, we refer to their ability to: communicate with each other using an expressive communication language; work together cooperatively to accomplish complex goals; act on their own initiative; and use local information and knowledge to manage local resources and handle requests from peer agents. Languages that facilitate high-level communication are thus an essential component of intelligent software agent architecture.

2.1. Agent Communication Languages

Most Agent Communication Languages (ACL) are based on the speech-act theory. Speech acts are expressed by means of standard keywords, also known as 'performatives' (for example 'ask', 'request' or 'tell'). The agent's message can then include several parameters such as 'sender' and 'receiver' of the message, the 'language' and 'ontology' (vocabulary) of the embedded content, and the actual 'content'. One can also add references to maintain the message history.

3. Basic Concepts of ACL

ACL provides agents with a means of exchanging information and knowledge. Michael R. Genesereth has gone as far to equate agency with the ability of a system to exchange knowledge using an ACL. Of course, other means have been used to achieve the lofty goal of seamless exchange of information and knowledge between applications. From remote procedure call and remote method invocation (RPC and RMI) to CORBA and object request brokers, the goal has been the same. What distinguish ACLs from such past efforts are the objects of discourse and their semantic complexity. ACLs stand a level above CORBA for two reasons:

- ACLs handle propositions, rules, and actions instead of simple objects with no semantics associated with them.
- An ACL message describes a desired state in a declarative language, rather than a procedure or method.

But ACLs by no means cover the entire spectrum of what applications might want to exchange. Agents can and should exchange more complex objects, such as shared plans and goals, or even shared experiences and long-term strategies. At the technical level, when using an ACL, agents transport messages over the network using a lower-level protocol—for example, SMTP, TCP/IP, IIOP, or HTTP. The ACL itself defines the types of messages (and their meanings) that agents can exchange. Agents do not, however, just engage in single-message exchanges; they have *conversations*—task-oriented, shared sequences of messages that they follow, such as a negotiation or an auction. At the same time, a higher-level conceptualization of the agent's strategies and behaviors drives the agent's communicative (and non -communicative) behavior.

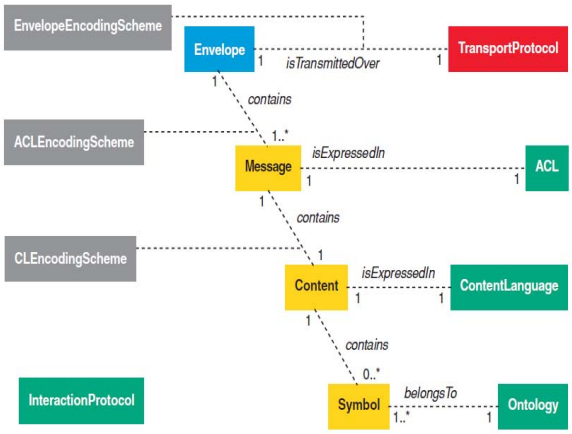

Fig 2: FIPA Standard Component of the communication language [2]

4. ACL Requirements

In this section we identify requirements for agent communication languages. We divide these requirements into seven categories: form, content, semantics, implementation, networking, environment, and reliability. We believe that an agent communication language will be valuable to the extent that it meets these requirements [1].

4.1. Form

A good agent communication language should be declarative, syntactically simple, and readable by people. It should be concise, yet easy to parse and to generate. To transmit a statement of the language to another agent, the statement must pass through the bit stream of the underlying transport mechanism. Thus, the language should be linear or should be easily translated into a linear stream of characters.

4.2. Content

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. The language should commit to a well-defined set of communicative acts (primitives).

4.3. Semantic

Semantics is an issue that has often been neglected during the design of communication languages. Such neglect is the direct result of the obscurity that surrounds the purpose and the desired features of communication languages. Although the semantic description of communication languages and their primitives is often limited to natural language descriptions, a well-defined semantic description is anything but a luxury. This is especially true if the communication language is intended for interaction among a diverse range of applications. Applications designers should have a shared understanding of the language, its primitives and the protocols associated with their use, and abide by that shared understanding.

4.4. Implementation

The implementation should be efficient, both in terms of speed and bandwidth utilization. It should provide a good fit with existing software technology. The interface should be easy to use; details of the networking layers that lie below the primitive communicative acts should be hidden from the user. It should be easy to integrate or build application program interfaces for a wide variety of programming languages, including procedural languages (e.g., C and Lisp), scripting languages (e.g., Tcl and Perl), object-oriented languages (e.g., Smalltalk and C++), and logic programming languages (e.g., Prolog).

4.5. Networking

An agent communication language should fit well with modern networking technology. This is particularly important because some of the communication will be about concepts involving networked communications. The language should support all of the basic connection types--point-to-point, multicast

and broadcast. Both synchronous and asynchronous connections should be supported. The language should contain a rich enough set of primitives that it can serve as a substrate upon which higher-level languages and protocols can be built. Moreover, these higher-level protocols should be independent of the transport mechanisms (e.g., TCP/IP, email, http, etc.) used.

4.6. Environment

The environment in which software agents will be required to work will be distributed, heterogeneous, and dynamic. To provide a communication channel to the outside world in such an environment, a communication language must provide tools for coping with heterogeneity and dynamism. It must support interoperability with other languages and protocols. It must support knowledge discovery in large networks. Finally, it must be easily attachable to legacy systems.

4.7. Reliability

A communication language must support reliable and secure communication among agents. Provisions for secure and private exchanges between two agents should be supported. There should be a way to guarantee authentication of agents. Because neither agents nor the underlying transport mechanisms are infallible, a communication language must be robust to inappropriate or malformed messages. The language should support reasonable mechanisms for identifying and signaling errors and warnings.

5. The KQML Language

Knowledge Query and Manipulation Language (KQML) is a language that is designed to support interaction among intelligent software agents. It was developed by the ARPA-supported Knowledge Sharing Effort and independently implemented by several research groups. It has been successfully used to implement a variety of information systems using different software architectures.

Communication takes place on several levels. The content of a message is only a part of the communication. Locating and engaging the attention of another agent with which an agent wishes to communicate is a part of the process. Packaging a message in a way that makes clear the purpose of an agent's communication is another. When using KQML, a software agent transmits content messages, composed in a language of its own choice, wrapped inside of a KQML message. The content message can be expressed in any representation language, and can be written in either ASCII strings or one of many binary notations (e.g. network-independent XDR representations). KQML implementations ignore the content portion of a message except to recognize where it begins and ends.

The KQML language simplifies its implementation by allowing KQML messages to carry arbitrary useful information, such as the names and addresses of the sending and receiving agents, a unique message identifier, and notations by any intervening agents.

The syntax of KQML is based on a balanced parenthesis list. The initial element of the list is the performative; the remaining elements are the performative's arguments as keyword/value pairs. Because the language is relatively simple, the actual syntax is not significant and can be changed if necessary in the future. The syntax reveals the roots of the initial implementations, which were done in Common Lisp; it has turned out to be quite flexible.

A KQML message from agent joe representing a query about the price of a share of IBM stock might be encoded as:

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

In this message, the KQML performative is *ask-one*, the content is *(price ibm ?price),* the ontology assumed by the query is identified by the token *nyse-ticks*, the receiver of the message is to be a server identified as *stock-server* and the query is written in a language called *LPROLOG*. The value of the :content keyword is

the content level, the values of the :reply-with, :sender, :receiver keywords form the communication layer and the performative name, with the :language and :ontology form the message layer. In due time, stock-server might send to joe the following KQML message:

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

The standby performative expects a KQML expression as its content. It requests that the agent receiving the request hold the stream of messages and release them one at a time; the sending agent requests a reply with the next performative. The exchange of next/reply messages can continue until the stream is depleted or until the sending agent sends either a discard message (i.e. discard all remaining replies) or a rest message (i.e. send all of the remaining replies now). This combination is so useful that it can be abbreviated.

KQML has about two dozen reserved performative names, which fall into seven basic categories which is shown in Table 1.

Category	Name
Basic query	evaluate, ask if, ask-about, ask-one, ask-all
Multi-response (query)	stream-about, stream-all, eos
Response	reply, sorry
Generic informational	tell, achieve, cancel, untell, unachieved
Generator	standby, ready, next, rest, discard, generator
Capability-definition	advertise, subscribe, monitor, import, export
Networking	register, unregister, forward, broadcast ,route

Table 1: categories of performatives

6. FIPA ACL

FIPA's agent communication language, like KQML, is based on speech act theory: messages are actions or communicative acts, as they are intended to perform some action by virtue of being sent. FIPA ACL is superficially similar to KQML. Its syntax is identical to KQML's except for different names for some reserved primitives. Thus, it maintains the KQML approach of separating the outer language from the inner language. The outer language defines the intended meaning of the message; the inner, or content, language denotes the variety of interesting agent architectures. Thus, KQML introduces a small number of KQML performatives that agents use to describe the metadata specifying the information requirements and capabilities; it also introduces a special class of agents called *communication facilitators*. A facilitator is an agent that performs various useful communication services, such as maintaining a registry of service names, forwarding messages to named services, routing messages based on content, matchmaking between information providers and clients, and providing mediation and translation services.

7. Applications of KQML Language

KQML has been used as the communication language in several technology integration experiments in the ARPA Rome Lab Planning Initiative. One of these experiments supported an integrated planning and scheduling system for military transportation logistics linking a planning agent, with a scheduler (in Common Lisp), a knowledge base, and a case based reasoning tool (in Common Lisp). All of the components integrated were preexisting systems which were not designed to work in a cooperative distributed environment. In a second experiment, an information agent was constructed by integrating

CoBASE, a cooperative front-end, SIMS , an information mediator for planning information access, and LIM , an information mediator for translating relational data into knowledge structures. CoBASE processes a query, and, if no responses are found relaxes the query based upon approximation operators and domain semantics and executes the query again.

8. Acknowledgement

We would like to thank our teacher professor S Vaniya for providing the complete guidance in the area of agent theory and motivation to work in this area. We are also thankful to our organization for providing all required resources and financial needs. At last we would like to thanks all the staff member of our department for giving continuous encouragement for publishing the paper.

9. References

[1] Evaluation of KQML as an Agent Communication Language paper by James Mayfield Yannis Labrou ,Tim Finin. Computer Science and Electrical Engineering Department University of Maryland Baltimore County

Baltimore MD21228-5398USA

[2] JADE a White paper by F. Bellifemine, G. Caire, A. Poggi, G. Rimassa

[3] Agent Communication Languages: The Current

Landscape by Yannis Labrou, Tim Finin, and Yun Peng, University of Maryland, Baltimore County

[4] M.P.Singh. Towards a formal theory of communication for multiagent systems. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI '91),1991.

[5] Verifiable Semantics for Agent Communication Languages by Michael Wooldridge Queen Mary and Westfield College,Department of Electronic Engineering,London E1 4NS, United Kingdom

M.J.Wooldridge@qmw.ac.uk

[6] Trends in agent communication language by B.chaib-draa , laval university ,Canada .F.Dignum Utrecht university, The Netherlands.

[7] P. R. Cohen and H. J. Levesque. Communicative actions for artificial agents. In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), pages 65–72, San Francisco, CA, June 1995.

[8] P. R. Cohen and C. R. Perrault. Elements of a plan based theory of speech acts. Cognitive Science, 3:177–212, 1979.

[9] The Foundation for Intelligent Physical Agents. FIPA ’97 Specification part 2—Agentcommunication language. The text refers to the specification dated 10th October 1997.

[10] M.R.Genesereth, A.M. Keller, and O.M.Duschka, “Infomaster: An Information Integration System,” Proc. ACM Sigmod Int’l Conf. Management of Data,ACM Press, 1997.