International Journal of Computing Academic Research (IJCAR)
ISSN 2305-9184 Volume 4, Number 1(February 2015), pp. 19-26

http://www.meacse.org/ijcar

# Updates in Knowledge Query Manipulation Language for Complex Multiagent Systems

Ankit Jagga[1], Dimple Juneja[2] and Aarti Singh[3]

[1&3] MM Institute of Computer Technology & Business Management, Maharishi Markandeshwar University, Mullana, India

[2] DIMT, Kurukshetra (Haryana), India

## Abstract

Knowledge Query Manipulation Language (KQML) is a language that facilitates communication and interoperability among coordinating software agents. The existing specifications of KQML focus on perspective, meaning, syntax, semantics, coverage and context of communication to lead to the final result derived from the communication. It is desired that the new extension should support the abstract interaction among software agents coordinating in multi-agent systems. Further, literature reveals that standards that are implementation independent are also lacking. Therefore, the language which is normative and can make communication between heterogeneous agents operating cross-platforms compatible has always been the area of interest to scientific community. In particular, this premise of this work is to extend pragmatic component of KQML which would shore up the use of language as a protocol. Also, the implementation prototype of the proposal is being presented.

**Keywords:** Multiagent Systems, Knowledge Query Manipulation Language, Specifications, Pragmatics, Semantics, Performative

## 1. Introduction

Modern multiagent systems are basically knowledge based systems that involve multiple interacting agents which are autonomous entities responsible for performing a task within a system and also respond to messages proactively. Now, since various homogenous and heterogeneous agents are required to coordinate and cooperate to achieve a desired goal, therefore, these agents must agree to certain rules not only restricted at the interface levels but also at the end user application level. Primarily, cooperating agents should minimally agree on rules pertaining to sending and receiving the messages (Transportation); meaning of individual messages (Semantics); structure of conversations (Syntax) and architecture of systems i.e. rules related to connecting systems in accordance with constituent protocols. In order to meet the above mentioned requirements, KQML was developed to support communication amongst autonomous and asynchronous software agents and but initially it was concerned with the transport and language levels only. In fact, KQML establishes communication among agents through attributes such as querying, stating, believing, requiring, achieving, subscribing, and offering [1]. Eminent researcher Finn and his team [2] argue that "KQML should be defined as more than a language with syntax and semantics but must also include a protocol that governs the use of language i.e. the pragmatic component". The current work finds the motivation from this argument and an in-depth grilling of literature revealed the ample scope of improvement in existing specifications of KQML and hence an extended KQML is proposed in the upcoming sections.

The paper is structured as follows: Section 1 provided an overview of need of KQML and motivation behind the current work. Section 2 presents the background and existing specifications of KQML in detail. Section 3 presents the proposed inclusions of novels pragmatic component including new performatives. Section 4 discusses the modified version of KQML with respect to a case study and section 5 finally concludes.

## 2. Background

Multiagent Systems (MASs) are complex systems in which various agents are required to communicate and coordinate in variety of domains including organization decision making. The inter-agent communication is found to be extremely significant during transfer of high level information as well as during negotiation in social systems. KQML was developed to establish this inter-agent communication which is independent of software architectures on which MASs are implemented. It is well suited to agent based complex systems as such systems are autonomous and also it is both message format and message handling protocol. Though, MASs are usually complex systems where agents operate at several levels, KQML interactions are classified to happen at three levels only i.e. Content Level, Message Level and Communication Level. While the content level depicts the knowledge and expression of the message to be transferred, the message level adds attributes such as language, ontology and speech act of message itself. Finally communication level adds few more attributes such as identity of sender and recipient and also the type of message i.e. synchronous or asynchronous.

A KQML message is called a performative where the defined performative is required to perform an action as desired by the sender. In fact, KQML is based on an extendable set of performatives, where performatives define the possible keywords for instituting interaction amongst KQML agents. It is only performatives that are responsible for identifying the protocol required to transfer a message and also provide the permissible speech-acts which sender can use to append with the message. Since, the aim of current work is to extend KQML by adding new performatives, therefore the scope of this paper is being limited to addition of few novel parameters to the existing set of performatives in KQML. Next subsection throws light on the existing performatives.

**2.1 Existing Performatives :** Before understanding the formal syntax and performative component of KQML, consider a situation where an *agent 1* wants to know about the operating system of other platform and hence sends a request to *agent2* operating in the respective environment and in response receives a reply as depicted in Fig. 1.

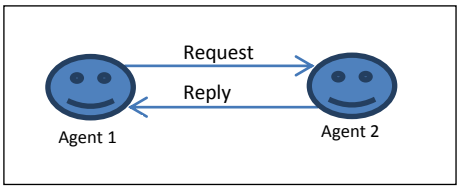


Fig. 1 : A Simple Agent Interaction

Fig. 2 represents the syntactic structure of above simple interaction as a KQML message.

```
(ask-one
:sender         agent 1
:content        (InfoEnv?info)
:receiver       agent 2
:reply-with     info-about-env
:language       Java
:ontology       Operating System
)
```

Fig. 2: A KQML Request

Table 1 delineates the semantics and functionality of query given in Fig. 2. In response to above request *agent 2* sends the reply to *agent 1* as given in Fig. 3.

```
(tell
:sender          agent 2
:content         (Info Env Windows)
:receiver        agent 1
:in-reply-to     info-about-env
:language        Java
:ontology        Operating System
)
```

Figure 3: The KQML Response

**Table 1:** Semantics and Functionality of KQML Query given in Fig. 2

| **Query Keywords** | **Semantics and Functionality** | **KQML Level** |
|---|---|---|
| *ask-one* | A KQML performative indicating the beginning of the request message | |
| *:sender* | Sender of query asking about the information about environment in which it may be required to execute ( *agent 1*) | communication level |
| *:receiver* | the receiver of the message required to send reply (*agent 2)* | |
| *:reply-with* | Reply with desired information (*info-about-env)* | |
| *:content* | the content i.e. the actual message *(Info Env?info)* | content level |
| *:ontology* | the ontology assumed to be known by all operating agents (*operating system)* | message level |
| *:language* | the language of query representation (JAVA) | |

The above response simply binds the name of operating system and returns the response to message with identifier *info-about-env*. It is worth mentioning that keywords shown above are reserved keywords only and in totality, there exist 36 KQML performatives [3] to facilitate the conversation between sender and receiver agent. These performatives belongs to three different domains [4,5] as shown in Fig. 4 and table 2 given below lists all performatives falling into these domains.

Since KQML facilitates interagent communication including negotiation, it incurs significant overhead in terms of time and bandwidth consumption and also transfer of low level data becomes an issue.

**Table 2: List of Existing Performatives in KQML [3]**

| **Sr. No.** | **Performative** | **Associated Meaning** |
|---|---|---|
| 1. | ask-if | S wants to know if the :content is in R's VKB |
| 2. | ask-all | S wants all of R's instantiations of the :content that are true of R |
| 3. | ask-one | S wants one of R's instantiations of the :content that is true of R |
| 4. | stream-all | multiple-response version of ask-all |
| 5. | eos | the end-of-stream marker to a multiple-response (stream-all) |
| 6. | tell | the sentence is in S's VKB |
| 7. | untell | the sentence is not in S's VKB |
| 8. | deny | the negation of the sentence is in S's VKB |
| 9. | insert S | asks R to add the :content to its VKB |
| 10. | uninsert | S wants R to reverse the act of a previous insert |
| 11. | delete-one | S wants R to remove one matching sentence from its VKB |
| 12. | delete-all | S wants R to remove all matching sentences from its VKB |
| 13. | undelete | S wants R to reverse the act of a previous delete |
| 14. | achieve | S wants R to do make something true of its physical environment |
| 15. | unachieve | S wants R to reverse the act of a previous achieve |
| 16. | advertise | S wants R to know that S can and will process a message like the one in a :content |

| | | |
|---|---|---|
| 17. | unadvertise | S wants R to know that S cancels a previous advertise and will not process anymore messages like the one in the :content |
| 18. | subscribe | S wants updates to R's response to a performative |
| 19. | error | S considers R's earlier message to be malformed |
| 20. | sorry | S understands R's message but cannot provide a more informative response |
| 21. | standby | S wants R to announce its readiness to provide a response to the message in:content |
| 22. | ready | S is ready to respond to a message previously received from R |
| 23. | next | S wants R's next response to a message previously sent by S |
| 24. | rest | S wants R's remaining responses to a message previously sent by S |
| 25. | discard | S does not want R's remaining responses to a previous(multi-response) message. |
| 26. | register | S announces to R its presence and symbolic name |
| 27. | unregister | S wants R to reverse the act of a previous register |
| 28. | forward | S wants R to forward the message to the :to agent ( R might beThat agent) |
| 29. | broadcast | S wants R to send a message to all agents that R knows of |
| 30. | transport-address | S associates its symbolic name with a new transport address |
| 31. | broker-one | S wants R to find one response to a <performative>(some agent |
| 32. | broker-all | S wants R to find all responses to a <performative>(some agentother than R is going to provide that response) |
| 33. | recommend-one | S wants to learn of an agent who may respond to a <performative> |
| 34. | recommend-all | S wants to learn of all agents who may respond to a <performative> |
| 35. | recruit-one | S wants R to get one suitable agent to respond to a <performative> |
| 36. | recruit-all | S wants R to get all suitable agents to respond to a <performative> |

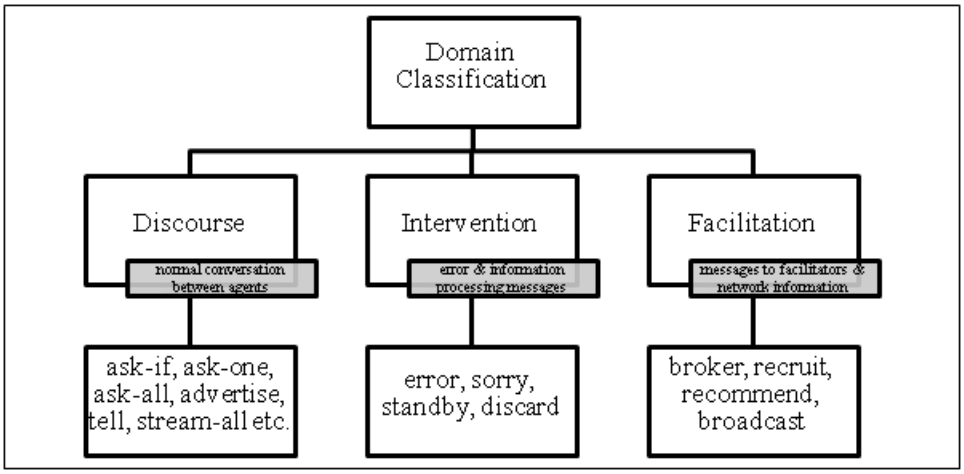


Fig. 4: Classification of Performatives Domain

Literature presented in the upcoming section highlights the research done in the ACL domain and also unfolds the issues to be addressed in future.

## 3. Literature Review

Extensive research has been done in the area of Agent Communication Languages(ACLs). This section highlights the work of eminent researchers and explores the challenges.

An in depth survey comparing the pros and cons of various ACLs and protocols is given in [6]. Researchers [7] have been continuously putting efforts to improve existing ACL according to FIPA standards and also carrying out the task cooperatively to achieve the shared goals. Authors in [8,9,10] have addressed the issue of semantics for KQML, in particular. They described KQML a language and associated protocol by which intelligent software agents can communicate to share information and knowledge. They believed that KQML, or something very much like it, will be important in building the distributed agent-oriented information systems of the future.

Another work by Vaniya et.al. [11] addressed the issuesof KQML in particular. Covington [12] examined the encoding of speech acts in KQML, contrasts KQML withhuman speech and conventional EDI; and suggested ways of improving KQML.

As mentioned in the previous section that only limited number of performatives have been defined so far which lays the foundation of agent communication in a MAS. In fact, very few perforomatives related to security of messages in KQML exists. Very recently [13] addressing the security need have been proposed but credibility of the proposed model has not been proved. Further, mechanisms related to finding the state of agent are also lacking. Hence, there is an ample scope to improve the existing set of KQML. It can be improved either by adding new performatives, adding new parameters or creating new ontologies. Further, a mechanism for creating machine-readable performative definitions is also desirable. Since, it is not possible to address all above mentioned shortcomings, this paper focuses on adding new performatives and rest may be taken up as future work.

## 4. Proposed Extensions

It is a well observed fact that the modern multiagent systems are not only complex but also demands a quick and a timely response. The list of KQML performatives does not offer any such performatives or parameters that address this issue and hence, we propose to introduce new parameters that would not only consider timing constraints but also would consider priority and quality of service being offered during that period.

Considering the above requirement, we analyzed the communication happening in a MAS and it was observed that primarily the sender and receiver agent desires that the request and reply from one agent to another shall be delivered well within time. Also, the advertisements about the capabilities of service providers shall be updated immediately or at least within time constraints. This section proposes few new parameters to be used in the envelope of existing performatives. These parameters pertain to priority of the request, timings and quality of response, in particular.

For instance, a user expects that all fluctuations of stock market should be conveyed on high priority and well within time limits. A KQML query for such a conversation of *user_agent* with *stock_market_server* would be represented as shown in Fig. 4.

```
(ask-one
:sender          user_agent
:content         (Valueof Stock ?value)
:receiver        stock_market_server
:reply-with      value_of_stock
:language        Java
:ontology        Stock Market
:priority        (por=high, qor=high, exe=now)
                 (por=mod, qor=avg/low, exe=whenever)
                 (por=low, qor=avg/low, exe=whenever/never)
)
```

Fig. 5: A KQML Request with NewPriority Parameter

As shown in Fig. 4, a *:priority* parameter has been added to KQML *ask-one* performative to express the priority of request (*por*) along with quality of response (*qor*) and constraints pertaining to the time of

execution (*exe)* of the task. The above example indicates that the *stock_market_server* now have the option of responding in three different modes as per the data structures shown in table-3.

**Table 4: Data Structures Associated with Priority Parameter**

| priority of request (*por*) | quality of response (*qor*) | time of execution (*exe)* |
|---|---|---|
| High | High | Now |
| Moderate | Average/Low | Whenever |
| Low | Average/Low | Never |

Further, *capabilities* parameter related to advertisement of capabilities can be included in *advertise* performative. In a MAS, all service providing agents must advertise their capabilities to all other agents well within a time limit else the services may get outdated without being utilized. Also, agents can advertise their potential timings to execute the query and respond. For instance, a search agent (service provider) may be able to search for english novels in 0.6secs on Google while the same agent takes 0.5secs on Yahoo compromising on the quality of response. The KQML query for such a response would be written as shown in Fig. 5.

```
(advertise
:sender          search_agent
:content         (English Novels?info)
:receiver        user_agent
:reply-with      info_about_english_novels
:language        XML
:ontology        Books
:capabilities    (timings=0.6 , qor=high)
                 (timings=0.5 , qor=low)
)
```

Fig. 6: A KQML Request with Capabilities Parameter

In this example, the search engine specifies through capabilities parameter that it has two execution strategies i.e. one can execute in 0.6 secs with high quality of response while the other can execute in 0.5 seconds with low quality of response. The capabilities parameter is used to express the quality of response as well as timing characteristics of response to be generated.

The proposed additionof *priority* parameter in the ask-one KQML performative considers the priority of request, quality of response and time of execution constraints that the user inputs for the particular message. These constraints can then be taken into account by the scheduling algorithm which is to be unified later. The scheduler determines the scheduling priority and execution time values for each agent request depending on the constraint values specified by the user.

The *capabilities* parameter added to the advertise performative allows an agent communicating with the service provider to specify the execution time as well as the quality of the result returned by each execution strategy of that agent.

### 4.1 Implementation Prototype

On the basis of above discussed background and literature, it can be stated that the agent communication is only possible through KQML performative method. Now, in order to implement the parameters added to performatives, a framework for parsing the performatives is required. The communication between two agents as seen by a user is an abstract view and is treated as virtual communication path. However; the actual communication is required to go through a series of steps as depicted in Fig. 6. This implementation prototype is based on the architecture for Real Time Multiagent Systems (RTMAS) [14] that has the capability to express and enforce QoS constraints. Although, our architecture is still under development but it is based on real-time CORBA infrastructure that will relax us from worrying out low-level inter-agent communication.

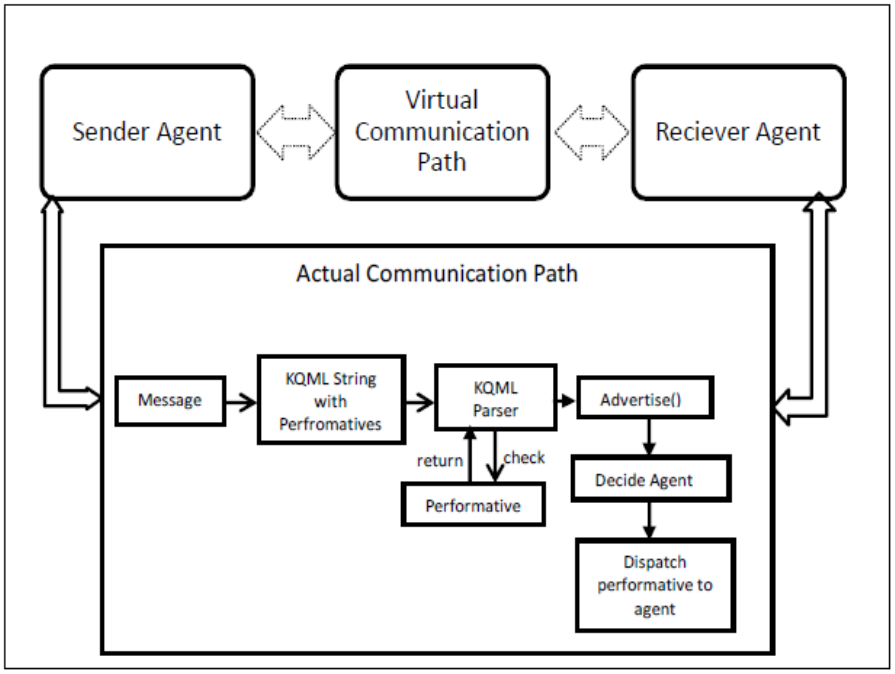


Fig. 7: Implementation Prototype

Following is the sequence of steps executing during actual communication path:

Step 1: Sender agent is required to send a message to receiver agent. The message is converted to KQML string with performatives based on ASCII.

Step 2: The converted string is submitted to KQML parser for parsing the tokens, parameters and performatives.

Step 3: The parser checks for received performatives and calls for advertise() routine. The advertise() routine contains ids of all agents with their capabilities.

Step 4: On the basis of performative, advertise routine decides the agent to be called for executing the string and dispatch the performative to desired agent.

Step 5: Receiving agent checks all the parameters, executes the message and returns response through the same path executed.

On the basis of above prototype, it can be observed that introduction of new parameters do not demand for new underlying infrastructure and hence the initial design of above given prototype is based on the implementation of the KCobalt system [15] that maps KQML messages to CORBA IDL.

## 5. Conclusion

The paper began with the exploring the feasibility of extending KQML and also presented research directions towards extending KQML. A brief background of existing specifications of KQML was also described and later a proposal regarding new parameters added to existing performatives was presented. An implementation prototype was explained and the implementation is the same is in progress.

## References

[1] Unisys, Software User Manual for KQML (Knowledge Query Manipulation Langugae)m revision 3.0, March 1995, www.csee.umbc.edu/csee/research/kqml/software/kats/kqml-sum.ps

[2] Tim Finin, Richard Fritzson, Don McKay, " An overview of KQML: A Knowledge Query Manipulation Language) ", A Technical Report, 1992, http://citeseerx.ist.psu.edu/showciting?cid=1105909

[3] Y.Labrou and T.Finin, A Proposal for a New KQML Specification, Tech.Report TRCs-97-03, computer Science and Electrical Engineering Dept., Univ. of Maryland,Baltimore County, baltimore, Md., 1997.

[4] Y.Labrou and T.Finin, A Proposal for a New KQML Specification, Tech.Report TRCs-97-03, computer Science and Electrical Engineering Dept., Univ. of Maryland,

[5] Baltimore County, baltimore, Md., 1997. Tim Finin, Don McKay, Rich Fritzson, and Robin McEntire. KQML as an Agent Communication Language. In The Proceedings of the Third International Conference on Information and Knowledge Management (CIKM), ACM Press, November 1994.

[6] Kalaivaini Subramaniam, ”Agent Communication Languages and protocols”, Departmentof Electrical and Computer Engineering, Universityof Calgary 2002.

[7] Moamin Ahmed, MohdSharifuddin Ahmad, MohdZalimanMohdYusoff, “A Review and Development of Agent Communication *Language”*, Published in Electronic Journal of Computer Science & Information Technology, VOL. 1 No. 1, May 2009 Universiti Tenaga Nasional Kajang , Malaysia.

[8] YannisLabrou, and Tim Finin “Semantics for an Agent Communication *Language”*Computer Science and Electrical Engineering Department, University of Maryland Baltimore County Baltimore MD 21250 USA.

[9] Tim Finin, Richard Fritzson, Don McKay and Robin McEntire “KQML as an Agent Communication Language”, University of Maryland Baltimore County Baltimore MD USA.

[10] YannisLabrou and Tim Finin*“*History, State of the Art and Challenges for Agent Communication Languages*”,* Department of Computer Science and Electrical Engineering University of Maryland, Baltimore County Baltimore, MD 21250.

[11] SandipVaniya, Bhavesh Lad and ShreyanshBhavsar “A Survey on Agent Communication Languages*”,* International Conference on Innovation, Management and Service, *IPEDR* vol.14(2011) Singapore 2011.

[12] M.A Covington, “Speech Acts, Electronic Commerce, and *KQML*” Decision Support Systems, 22 (3) (1998), pp. 203–211.

[13] Dimple Juneja, Aarti Singh, Renu Hooda, “A Three Tier Secure KQML Interface WithNovel Performatives”, International Science Index, vol. 8, No. 6, Part VI, pp. 788-792.

[14] Lisa Cingiser DiPippo, Victor Fay-Wolfe, Lekshmi Nair, Ethan Hodys, Oleg Uvarov, A Real-Time Multi-Agent System Architecture for E-Commerce Applications, Proceedings. 5th International Symposium on Autonomous Decentralized Systems, 2001, pp. 357 – 364

[15] D.Benech, T.Desprats. A KQML-CORBA based Architecture for Intelligent Agents Communication in Cooperative Service and Network Management. In Proceedings of IFIP/IEEE International Conference on Management of Multimedia Networks and Systems ’97 July 1997.