IMPLEMENTING KQML PERFORMATIVES FOR OUTSOURCING MANAGEMENT SYSTEM

P. Felcy Judith
Research Scholar
Mother Theresa Women’s University
Kodaikanal, Tamilnadu, India
felcy_judith@yahoo.com

Dr. E. Kirubakaran
Sr.DGM, Outsourcing
BHEL
Trichy, Tamilnadu, India
e_kiru@yahoo.com

Abstract—Agents are autonomous, social, pro-active, reactive, and communicate to achieve common goal. The Knowledge Query Manipulation Language (KQML) is essentially a knowledge-level messaging language developed in the context of the knowledge sharing between agents. KQML defines various performatives to enable agent communicate their intentions by sending messages. This paper focuses on implementing KQML performatives in .NET C# language, to enable the Outsourcing Management System (OMS) agents to communicate between themselves.

Keywords-Agent; Multi-Agents; Agent Oriented Software Engineering; Agent Communication; KQML;

I. INTRODUCTION

“An agent is a computer system, situated in some environment that is capable of flexible autonomous action in order to meet its design objectives” this is the commonly accepted definition for agents. Agents are flexible by exhibiting the following characteristics

Reactive – Agents were able to perceive their environment and respond to the changes occur in their environment in order to satisfy their design behavior

Pro-Active – Agents exhibit goal-oriented behavior by taking initiative in order to satisfy their design behavior

Being Social – Agents communicate with other agents in order to satisfy their design behavior

“Multi-agent systems (MAS) are computational systems in which a collection of loosely coupled autonomous agents interact in order to solve a given problem.” Agents communicate, cooperate, coordinate and negotiate for

a) Achieving a common goal
b) Monitor the progress of the team effort
c) Help another in need
d) Co-ordinate individual action so that they do not interfere with one another
e) Communicating success and failures
f) To establish zero competition among team members

In this paper we focus on agent communication by implementing KQML performatives in .NET C#, to enable agents to communicate.

Chapter II introduces the speech act theory and KQML, chapter III focus on Modeling Outsourcing Management System. Chapter IV focuses on implementing KQML performatives.

II. THE SPEECH ACT THEORY AND KQML

The Speech-act theory conveys language is action; an agent communicates and changes the state of the world. Thus the speech act changes the beliefs, desires and intentions of agents. Speech acts are classified according to their illocutionary force which means the ‘type’ of utterance.

The speech act generally contains two components

- performative verb – such as request, inform
- propositional content – to perform something

The KQML is a language and protocol for communication among software agents and knowledge-based systems. The KQML message format and protocol can be used to interact with an intelligent system, either by an application program, or by another intelligent system. KQML’s ”performatives” are operations that agents perform on each other’s knowledge and goal stores. The general form of the pre-defined internal action for communication is:

\[ .\text{send}(\text{receiver}, \text{illocutionary\_force}, \text{propositional\_content}) \]

The Table I list’s all available performatives, where s denotes the sender and r denotes the receiver

<table>
<thead>
<tr>
<th>TABLE I.</th>
<th>KQML PERFORMATIVES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layers</td>
<td>Description</td>
</tr>
<tr>
<td>tell</td>
<td>s intends r to believe (that s believes) the literal in the message’s content to be true</td>
</tr>
<tr>
<td>untell</td>
<td>s intends r not to believe (that s believes) the literal in the message’s content to be true</td>
</tr>
<tr>
<td>achieve</td>
<td>s requests r to try and achieve a state of affairs where the literal in the message content is true (i.e. s is delegating a goal to r)</td>
</tr>
<tr>
<td>unachieve</td>
<td>s requests r to drop the goal of achieving a state of affairs where the message content is true</td>
</tr>
<tr>
<td>askOne</td>
<td>s wants to know if the content of the message is true for r (i.e. if there is an answer that makes the content a logical consequence of r’s belief base, by appropriate substitution of variables)</td>
</tr>
<tr>
<td>askAll</td>
<td>s wants all of r’s answers to a question</td>
</tr>
<tr>
<td>tellHow</td>
<td>s informs r of a plan (s’s know-how)</td>
</tr>
<tr>
<td>untellHow</td>
<td>s requests that r disregard a certain plan (i.e. delete that plan from its plan library)</td>
</tr>
<tr>
<td>askHow</td>
<td>s wants all of r’s plans that are relevant for the triggering event in the message content</td>
</tr>
</tbody>
</table>
III. OUTSOURCING MANAGEMENT SYSTEM MODEL

Outsourcing management system is modeled for an organization which manages numerous vendors, who provide finished parts for the buyer organization, who internally use these products as raw materials in their manufacturing process. The buyer organization generally delegates project orders to the vendors along with necessary materials and technical specification to produce the needed product.

Vendor information service manages eligible vendor. Initially vendor provides information to the buyer about their infrastructure, experience and various others and later buyer processes this information and does necessary inspection and adds the eligible vendor to their vendor information system. The purchase auction service identifies the best deal by evaluating the vendor performance and other metrics. After identifying the right vendor the buyer organization provides purchase order along with necessary materials for production. The materials are managed by material management system of the buyer. In every phase of the project the buyer inspects the vendor. The inspection services are managed by the buyer organization, they appoint people from private agency or quality inspectors of their organization for inspection. After the production is declared complete the vendor could use transportation service hosted by the buyer to book people and resource to transport the produced good to buyer organization. The buyer organization entry formalities were managed by this service. Later through the payment service hosted by the buyer the project cost will be paid to the vendor. The complete financial process is managed by the payment service.

Figure 1 shows the goal model for outsourcing management system, it consists of indentify alternatives, find deal, ordering, material management, receiving finished product and paying. Vendor information service manages eligible vendor. Initially vendor provides information to the buyer about their infrastructure, experience and various others and later buyer processes this information and does necessary inspection and adds the eligible vendor to their vendor information system.

The purchase auction service identifies the best deal by evaluating the vendor performance and other metrics. After identifying the right vendor the buyer organization provides purchase order along with necessary materials for production. The materials are managed by material management system of the buyer. In every phase of the project the buyer inspects the vendor. The inspection services are managed by the buyer organization, they appoint people from private agency or quality inspectors of their organization for inspection. After the production is declared complete the vendor could use transportation service hosted by the buyer to book people and resource to transport the produced good to buyer organization. The buyer organization entry formalities were managed by this service. Later through the payment service hosted by the buyer the project cost will be paid to the vendor. The complete financial process is managed by the payment service.

Figure 1: GOAL MODEL FOR OUTSOURCING MANAGEMENT SYSTEM

Figure 2: GOAL MODEL FOR PURCHASE AUCTION SERVICE

Figure 3: GOAL MODEL FOR VENDOR SALES SERVICE

TABLE II. MOTIVATIONAL SCENARIO FOR BOTH BUYER ORGANIZATION PURCHASE PROCESS, VENDOR SALES PROCESS

<table>
<thead>
<tr>
<th>Scenario Name</th>
<th>Automated Procurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>The organization both buyer and vendor is represented by agents. The buyer organization needs some vendor to manufacture spares could be used for production of the buyer. As a process of finding the suitable vendor the buyer agent performs the following activities:</td>
<td></td>
</tr>
<tr>
<td>(a) Create RFQ</td>
<td></td>
</tr>
<tr>
<td>(b) Raise Enquiry</td>
<td></td>
</tr>
<tr>
<td>(c) Negotiate and Clarify Requirements</td>
<td></td>
</tr>
<tr>
<td>(d) Receive Technical / Price Quote from Vendors</td>
<td></td>
</tr>
<tr>
<td>(e) Decide the list of vendors for further negotiation</td>
<td></td>
</tr>
<tr>
<td>(f) Reverse Auction</td>
<td></td>
</tr>
<tr>
<td>(g) Negotiate with vendors</td>
<td></td>
</tr>
<tr>
<td>(h) Provide order for needed product</td>
<td></td>
</tr>
<tr>
<td>The seller organization delegates the following activities to their seller agent</td>
<td></td>
</tr>
<tr>
<td>(a) Receives RFQ</td>
<td></td>
</tr>
<tr>
<td>(b) Negotiate and clarify the specification needed</td>
<td></td>
</tr>
<tr>
<td>(c) Check Feasibility</td>
<td></td>
</tr>
<tr>
<td>(d) Send Quote</td>
<td></td>
</tr>
<tr>
<td>(e) Negotiate with buyer</td>
<td></td>
</tr>
<tr>
<td>(f) Receive Order</td>
<td></td>
</tr>
<tr>
<td>ISSN:2229-6093</td>
<td></td>
</tr>
</tbody>
</table>
Figure 2 gives the sub-goals of the purchase auction service which find the right vendor for the buyer organization. The buyer organization receives the needed quote from the vendor by raising enquiry by providing needed technical information to the vendor. Later it provides clarification for the queries raised by the vendor organization before receiving the quote. The entire communication between the buyer and vendor agent is through secure communication. This is one of the quality goals in this process. After receiving the necessary quote from the vendor(s) the technical and the price bids will be opened in stages. Based on the quality goals like “minimal price”, “vendor performance” and “schedule and other capabilities”, the one or more vendors will be selected for further communication. Later the buyer organization will do a reverse auction, negotiates with the vendor organization to choose the right vendor. Later the purchase order along with raw materials will be provided to the vendor to initiate production process.

Figure 3 gives the sub-goals of the vendor sales service which in turn reflects the vendor perspective of the OMS. The vendor organization receives the request for quote (RFQ) from the buyer along with the technical specification, delivery schedule. The vendor agent communicates with the buyer agent to clarify points in the specification and schedule. Later it evaluates the feasibility by considering the quality goals of “resource availability”, “schedule availability”, “cost analysis” and provides quote to the buyer agent. Once the buyer agent confirms the selection of the vendor for further negotiation, vendor agent participates in the negotiation and responds to schedule or cost of other changes requested by the buyer. Once receiving the purchase orders from the buyer it vendor agent transfers the order to other agents to enable post-order development activities.

The AOR behavior model given in figure 4 represents the computational behavior modeling viewpoint. This shows the complete communication between the vendor and buyer agent. Here the buyer agent communicates to achieve quote from vendor agent. The communication between buyer and vendor is asynchronous, and it can be either “tell” or “achieve” performative of KQML depends of if it waits for the reply or not. During negotiation for schedule or cost or other changes the performatives “askone”, “askall”, “tellhow” were executed. Later the vendor “acheive” or “unacheive” the order depends on the negotiation and the buyer order process consideration.

IV. IMPLEMENTING KQML PERFORMATIVES IN .NET C#

KQML performatives were discussed in chapter II. As discussed in chapter III the buyer and vendor agents communicate between each other in various occasions in various performatives. The performatives are broadly classified into

- Information exchange
- Goal delegation
- Information seeking

This chapter explains how these performatives are implemented in c#.

The above code explains how the buyer agents “tell” the vendor agent to provide quote by providing the schedule and specification. Here the buyer agent informs the need and returns immediately without waiting for the vendor agent to respond. The same message could be passed to multiple agents as information.

The above code explains how the buyer agent could “untell” the vendor agent to inform not to provide quote and undo the earlier belief state which requested to provide quote. The same message could be passed as information to multiple agents as well.
**B. Goal Delegation**

In contrast to information seeking performatives here the buyer agent waits for the reply of the vendor. Here the buyer agent delegates the goal e.g. GetQuote to the vendor agent and waits for the reply of the vendor. The vendor investigates and processes the request and if it cannot respond to the request due to internal state or insufficient privilege it can very well ignore the request. In other case if it is ready with the quote requested then it can callback the buyer e.g. ProcessQuote provided along with the request.

```csharp
m.SendRFQ(vendorAgent, MessageType.GetQuote,
   (o) => vendorAgent.GetQuote(
       new RFQ
       { Identifier = "134",
         ExpectedQuote = Schedule.GetQuote(),
           //quote => buyerAgent.ProcessQuote(quote)));
```

In the case of “unachieve” the buyer quotes the earlier. It is similar to “untell” but the difference is it drops the belief and stops the plan getting executed. Here care must be taken while stopping the plan when it is in execution.

```csharp
m.SendRFQ(vendorAgent, MessageType.Unachieve,
   (o) => vendorAgent.GetQuote(
       new RFQ
       { Identifier = "134",
         ExpectedSchedule = Schedule.GetSchedule(),
```

**C. Information Seeking**

In the above code the buyer agent as a part of negotiation need to make a change in schedule. The buyer agent communicates to the vendor asking the vendor agent to ask another agent e.g. ProductionAgent to check if the schedule could be altered. Here the vendor agent does not have the information directly it needs to communicate to the production agent to get whether the needed schedule could be achieved or not. The buyer here waits for the reply and once the vendor agent gets the reply it could callback the buyer e.g. ProcessChangeInSchedule by providing the suitable reply.

```csharp
m.SendRFQ(vendorAgent, MessageType.ChangeSchedule,
   (o) => vendorAgent.ChangeSchedule(
       buyerAgent.GetScheduleChange(),
       reply => buyerAgent.ProcessChangeInSchedule(reply)));
```

In this case of “askall” the buyer agent is interested in detailed reply from more agents. If the buyer needs clear replies from one or more agents in response to the change in schedule, it communicates to the vendor agent asking the agent to communicate to all the other agents need change in plan due to the change in schedule requested. The vendor waits for all the reply and later callback the buyer e.g. ProcessChangeInSchedule with list of agents and their replies.

```csharp
m.SendRFQ(vendorAgent, MessageType.Tellhow,
   (planQueue) => vendorAgent.GetDeliveryPlan(
       buyerAgent.GetDeliveryPlan( )))];
```

Finally here in “tellhow” the buyer communicates to the vendor for e.g. detail plan of delivery. Here buyer conveys the entire plan in a queue. This detailed plan gets updated to the vendor belief’s base. If the buyer expects a reply the vendor responds to the buyer with the provided callback. In the above code the callback is not provided and hence the vendor could stop after updating its belief’s base with the plan provided.

**V. CONCLUSION**

In this paper the speech act theory and KQML was analyzed. Later the outsource management system model was briefly introduced with goal models and various performative of communication between agents was explained with behavior model. Finally the implementation of various KQML messaging performative in .NET C# language was explained.

**VI. REFERENCES**


