An Agent Communication Language Using XML and RDF

Jangwon Choi, Yeongho Kim, Suk-Ho Kang
Department of Industrial Engineering, Seoul National University

Abstract

We propose a message-based communication protocol for software agents. An emphasis is placed on the problems of sharing and exchanging information through agent communication. We adopt the Resource Description Framework (RDF) schema for information modelling in business domain. The eXtensible Markup Language (XML) serialization is used to generate messages for agent communication. The use of XML and RDF enables software agents to understand the contents of messages correctly and consistently. We believe that the approach can provide a promising way to the automation of business processes through seamless communication among the partners.

1. Introduction

Recent development and deployment of Internet and agent technologies have increased the efficiency of information exchange in many application areas. As for the electronic commerce domain, much research work focuses on agent-based systems to facilitate various types of on-line auction (Sandholm 2000, Vulkan 2000, Cabri 2001, Yokoo 2001), negotiation for goods and services (Fischer 1999, Huhns 1999, Lee 2000, Collins 2001), brokering between buyers and sellers (Kwang 2000, Klusch 2001), and collaboration among business partners (Ulieru 2000, Huhns 2001, Wu 2001). Recently SAP, a leading ERP (Enterprise Resource Planning) and e-business solution providing company, announced an enhancement of its solution to manage adaptive supply chain networks through the use of intelligent agent technology. They expect an adaptive supply network will provide business partner integration and dynamic collaboration through portals and exchanges (SAP AG 2001). Although there are a number of agent-based information exchange systems to support collaboration among business partners via the Internet, there has been no dominant solution so far.

We propose an agent communication language which adopts the Resource Description Framework (RDF) schema for information modelling in specific application domain. The XML serialization is used to generate XML/RDF messages. The use of XML and RDF enables software agents to understand the contents of messages correctly and consistently. We believe that the approach can provide a promising way to the automation of business processes in many application domains through seamless communication among the partners.

2. Related technology

The key technologies employed in this research are XML, RDF, and agent. An overview of these is presented in this section.

2.1. XML

An important research issue in system integration is how to guarantee interoperability among different applications running on different hardware platforms. XML, a standard for exchanging electronic document by World Wide Web Consortium (W3C), is conceived as a promising alternative to solve the problem. It also enables platform-independent implementation of heterogeneous application systems. XML documents are platform-independent in themselves, and an application handling the XML documents can be developed in a standard form using Document Object Model (DOM) and Simple API for XML (SAX) (Nakhimovsky 2000). In particular, a Web-based application can be readily developed using a Java-based standard library. Another advantage of XML is that existing transfer protocols, such as Hypertext Transfer Protocol (HTTP) and Simple Object Access Protocol (SOAP), can be used directly and that it provides a serialization to transfer various data models through the protocols. In our approach, RDF models, described in the subsequent section, are serialized in XML, and exchanged among our system components.

2.2. RDF

RDF, also recommended by W3C, is a general framework to describe resources on the Web. It provides an abstract and conceptual metadata model for the resources, and serves as a basis for facilitating the semantic level information exchange among Internet users via shared schema of the metadata model (Lassila 1999, Brickley 2000). RDF schema can be defined by anyone and for any application domain. The RDF schema specification recommendation is found in Brickley (2000).

In order to communicate an RDF statement, a graph structure of RDF model needs to be expressed in a structured message format. This is called serialization. As mentioned before, we use the XML serialization. Readers may refer to Lassila (1999) for RDF syntax specifications and abbreviated syntax.

As for the interoperability, resources and properties need to be predefined in an RDF schema. The schema provides a unique identification of the resources and properties using XML namespace (the term `xmlns`). This is similar to the concept of ontology to ensure consistency in agent communication. The ontology specifies the terms that are exchanged and understood during communication. As one specifies such ontology, one can prevent agents from confusing about the meanings and valid attribute values of terms they use (Bradshaw 1997). We believe the XML and RDF schema are best fit for such purpose. This enables automated information exchange and interoperation among multiple software agents described in the next section.

2.3. Agent communication

A prerequisite to the agent communication is that all the participating agents should be able to understand the communication contents. This means that the agents should use the same language and ontology. The most popular language is Knowledge Query and Manipulation Language (KQML) (Bradshaw 1997). One problem in most previous agent systems employing KQML is that there are in fact several different types of KQML. Each system uses its own KQML for its internal use only, and thus the meaning of the language is differed depending on the context (Labrou 1999). Most previous agent-based systems employ an ontology server that assists agents to have the same meaning according to the context. Recently, there are researches focusing on the use of XML in agent communication (Glushko 1999, Makatchev 2000, Shiau 2000). They emphasize the advantages of XML in view of information exchange and standard structure for content representation. For the agent communication in this research, we have designed an Agent Communication Language (ACL) using XML, RDF, and KQML. Agents can communicate with a simple mechanism that allows sharing RDF schema through XML namespace. Table 1 compares the major communication methods in previous researches. Some of the comparison criteria are adopted from Singh (1998).

3. Message structure for agent communication

One requirement for the communication is that the interpretation of message contents be semantically equivalent over all related agents. Also important is to assure consistent processing of the same message. To assure these, we use XML, RDF, and KQML.

3.1. Message layers

We adopt the RDF schema to model the information exchanged among agents. In this section, a general structure of XML/RDF message is explained.

Figure 1 illustrates the message structure with an example. The message is composed of three layers: XML/RDF, KQML, and content layers.

The XML/RDF layer should include three

Table 1. Agent communication languages and their characteristics

Language Type / Criteria	KQML-based (Laufmann 1997)	XML Document-based (Glushko 1999)	KQML/XML-based (Makatchev 2000)	XML/RDF-based (This paper)
Message Type	KQML Message	XML Document	KQML/XML Message	XML/RDF Message
Human Readability	Low	High	High	High
Message Parser	Proprietary	XML Parser	(KQML) XML Parser	XML Parser
Semantic (Ontology) Expression	Proprietary	XML DTD	XML Schema	RDF Schema
Coverage in Performatives	Limited	N/A*	Limited	Expandable (RDF Schema)
Perspective	Private	Public	Private	Public
Type of Meaning	Personal	Personal	Personal	Conventional (RDF Schema)
Communicating Participants	Homogeneous	Heterogeneous	Homogeneous	Heterogeneous
Facilitation Type	Centralised	Centralised**	Centralised	Distributed (Interface Agent)
Design Autonomy	Low	High	Low	High

* : Exchanges business document forms as EDI systems do. Agents simply exchange and process the documents.

** : It is centralised in case of broker systems, such as auction and exchange. Otherwise, e.g. Peer-to-Peer, there is no facilitation.

namespaces for RDF syntax, KQML schema, and schema for certain context (we use PS (Product Support) schema here). The namespace for RDF syntax indicates that the message is an instance of RDF model, and thus it should be parsed using the RDF syntax. The other two namespaces are for KQML schema and PS schema, each of which is further described later.

Second, the KQML layer contains the performative used in a message and its associated attributes. In the example, the message uses the `ServiceRequest` performative, which should have the `sender`, `receiver`, `replyWith`, and `content` attributes. The performative means that the `sender` requests the `receiver` to provide a maintenance service for the facility specified in the `content`. The `content` attribute specifies the message identification that points the content in the next layer. The `replyWith` attribute indicates that the message receivers should reply to the message. The reply message should contain the `replyWith` attribute value, `HMC002`, so that the `sender` can identify it.

XML/RDF

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:kqml="http://cybernet.snu.ac.kr/kqml-schema#"
  xmlns:ps="http://cybernet.snu.ac.kr/product-support-schema#">
```

KQML

```
  <kqml:ServiceRequest about=" ">
    <kqml:sender resource="http://Xcorporation.com/agents/#TaskC001"/>
    <kqml:receiver
    resource="http://Ycorporation.com/agents/#Distributor37"/>
    <kqml:replyWith>HMC002</kqml:replyWith>
    <kqml:content>
      <kqml:Message ID="0001">
```

Content

```
        <ps:Facility about=" ">
          <ps:facilityID
          resource="http://Xcorporation.com/facility/#excavator12"/>
          <ps:install-location
          resource="http://Xcorporation.com/facility/#Plant5"/>
        </ps:Facility>
```

```
      </kqml:Message>
    </kqml:content>
  </kqml:ServiceRequest>
</rdf:RDF>
```

Figure 1. Structure of XML/RDF message

Finally, the content layer includes the subject matter of the message. It is determined depending on the performative described above. In our example, the content layer specifies the target object of maintenance and its location. The content specification should follow the content schema, `product-support-schema`, in the `ps` namespace. In the example, the `Facility` resource is identified by its two attributes `facilityID` and `install-location` of which the resource URLs are `http://Xcorporation.com/facility/#excavator12` and `http://Xcorporation.com/facility/#Plant5`, respectively.

The proposed method provides a standard message structure and syntax using XML, and allows an interchange of semantic level information using RDF. This assures a consistent interpretation and processing among different agents. All the agents refer the same namespace for RDF syntax, KQML schema, and PS schema, and thus they all have the same language syntax, communication protocol and content understanding. In the end, XML and RDF becomes the fundamental framework for information exchange among agents.

3.2. RDF schema for KQML layer

As described earlier, we use KQML-like performatives and attributes in the KQML layer. They determine an agents intention and behaviour in a message. We adopt some of the existing KQML performatives (Bradshaw 1997) and modify their names and meaning for our purpose.

We define an RDF schema model (KQML schema) to express and interpret the KQML performatives and attributes used in our prototype system. The KQML schema model is shown in figure 2. In the model, there is a high-level class `Performative`, which has three properties, `Sender`, `Receiver`, and `content`. All the other performatives are sub-classes of the high-level class, so that they all can inherit the common properties. In addition, each sub-class has its own properties, such as `inReplyTo` and `replyWith`. The model also shows that an agent (`Interface`, `Task`, and `Information Agent`) can become `Sender` and `Receiver`.

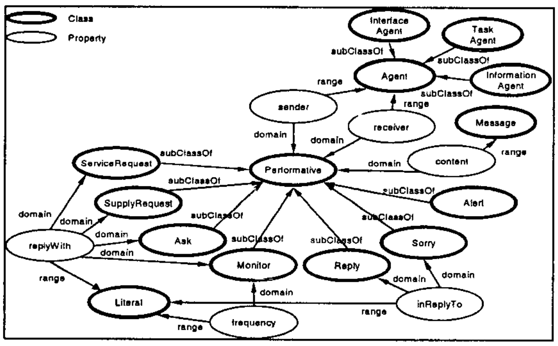

Figure 2. RDF schema model for KQML performatives

Since a message is written in the XML format (the XML namespace for KQML layer is `kqml:`), an XML parser alone can extract the performatives and attributes embedded in an XML/RDF message. That is, it is unnecessary to use any KQML parser, unlike other KQML-based systems. This is also true for the content layer described in the next section.

3.3. RDF schema for content layer

A content layer contains task-related information that is meaningful to an agent performing the task. We model the information again using an RDF schema (PS schema). Basically, an RDF schema consists of classes and related properties. The first step to develop a model for the content information is identifying resources and attributes in an application domain. Those can often be recognized by examining typical scenarios in the application domain. We use the Unified Modelling Language (UML) model, which is an object-oriented modelling tool for modelling and designing information systems.

In a UML model, the important part of scenario appears in use case diagrams. A diagram involves actors and tasks that the actors has to carry out. An example use case diagram for our product support logistics system is presented in figure 3. In this example, actors are customer, distributor, and manufacturer. Also included are some of the important activities in the product support logistics, such as request and provision of maintenance services and order and supply of service parts. In our system, agents carry out those activities on behalf of the actors.

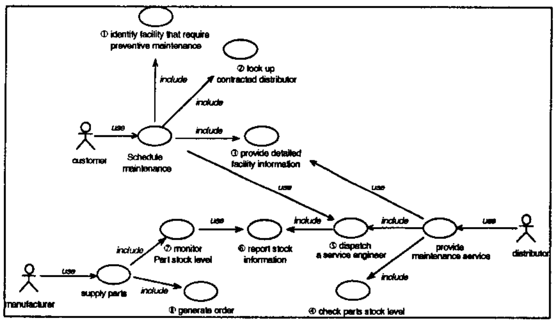

Figure 3. A use case diagram for product support logistics context

We describe the use cases while considering the agent actors and the activities that the agents should carry out. For example, consider the use case of service request/provision for preventive maintenance. The scenario listed below highlights the resources and their attributes (do not include all of them) involved in the service.

1) A customer identifies a <u>*facility*</u> that requires preventive *maintenance* based on its *maintenance schedule*.

2) The customer looks up a *contracted distributor* that can provide the <u>*maintenance service*</u>.

3) The customer requests the distributor of the service with detailed facility information that includes *manufacturer*, *facility ID*, *install-location*, *maintenance type*, and so on.

4) If the service involves <u>*part*</u> replacement, the distributor checks the availability of service parts with their *part IDs* in the distributors *warehouse*.

5) The distributor secures required parts and asks

a *service engineer* to perform the requested maintenance service.

6) When the service engineer takes the parts out of the distributors warehouse, the change in the *stock level* is reported to a *manufacturer*.

7) The manufacturer makes replenishment decision based on the stock level information.

8) A purchase *order* is generated on manufacturers side if the stock level is lower than the *reorder point*, and the parts are shipped to the distributors warehouse.

From such a scenario, we can identify resources and their attributes used in agents interaction. In the example, underlined italic words represent resources, and italic ones attributes.
We can use the resources and attributes to generate an RDF schema (PS schema). As mentioned above, an RDF schema consists of classes and their properties. In order to associate a resource and an attribute, we use `domain` and `range` constraint. The `domain` constraint assigns a resource to an attribute, and the `range` constraint assigns a value (this is a resource, too) to an attribute. The `subClassof` constraint defines a resource class that inherits from another class.

With the PS schema, an agent can generate and interpret the contents in a content layer. The contents involve the information relevant to a use case of preventive maintenance service. The PS schema allows that an agent identify a class and its associated properties automatically, so that it can assign a value to the property of a class instance. An RDF statement together with a KQML layer forms a complete XML/RDF message (Remind that a `Message` class gives a value of the `content` property in KQML layer). For example, in the XML/RDF message shown in figure 1, the agent that has generated the message expresses the intention that it *Requests preventive maintenance to Distributor37 for Excavator12 installed at Plant5.*

4. Conclusions

We believe that the proposed communication mechanism can be a basic framework of agent-based inter-company information exchange. The proposed method provides a standard message structure and syntax using XML, and allows an interchange of semantic level information using RDF. This assures a consistent interpretation and processing among different agents. In the end, XML and RDF becomes the fundamental framework for information exchange among agents.

References

Bradshaw, J. M., 1997, *Software Agents* (Menlo Park: AAAI Press/The MIT Press).

Brickley, D., et al,and Guha, R.V., 2000, Resource description framework schema specification. *W3C candidate recommendation.*

Cabri, G., Leonardi, L., and Zambonelli, F., 2001, Coordination infrastructures for mobile agents. *Microprocessors and Microsystems*, **25**, (2).

Collins, J., Bilot, C., Gini, M., and Mobasher, B., 2001, Decision processes in agent-based automated contracting. *IEEE Internet Computing*, **5**, (2).

Fischer, K., Chaib-draa, B., Muller, J. P., Pischel, M., and Gerber, C., 1999, A simulation approach based on negotiation and cooperation between agents: a case study. *Systems, Man, and Cybernetics, IEEE Transactions on*, **29**, (4).

Glushko, R. J., Tenenbaum, J. M., and Meltzer, B., 1999, An XML framework for agent-based e-commerce. *Communications of the ACM*, **42**, (3).

Huhns, M. N., and Malhotra, A. K., 1999, Negotiating for goods and services. *IEEE Internet Computing*, **3**, (4).

Huhns, M. N., and Stephens, L. M., 2001, Automating supply chains. *IEEE Internet Computing*, **5**, (4).

Klusch, M., 2001, Information agent technology for the Internet: A survey. *Data & Knowledge Engineering*, **36**, (3).

Kwang Mong Sim, and Chan, R., 2000, A brokering protocol for agent-based e-commerce. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, **30**, (4).

Lassila, O., and Swick, R. R., 1999, Resource description framework model and syntax specification. *W3C recommendation.*

Lee, K. J., Chang, Y. S., and Lee, J. K., 2000, Time-bound negotiation framework for electronic commerce agents. *Decision Support Systems*, **28**, (4).

Makatchev, M., and Tso, S. K., 2000, Human-robot interface using agents communicating in an XML-based markup language. *RO-MAN 2000, Proceedings of the 9th IEEE International Workshop.*

Nakhimovsky, A., and Myers, T., 2000, *Professional Java XML Programming* (Birmingham: Wrox).

Sandholm, T., 2000, Approaches to winner determination in combinatorial auctions. *Decision Support Systems*, **28**, (1-2).

SAP AG, 2001, SAP evolves supply chains into adaptive supply chain networks. http://wwwext03.sap.com/usa/sapphire/media/press/.

Shiau, J. Y., Ratchev, S. M., and Valtchanov, G., 2000, Distributed collaborative design and manufacturability assessment for extended enterprise in XML-based agent system. *WET ICE 2000, Proceedings of the 9th IEEE International Workshop.*

Singh, M. P., 1998, Agent communication languages: rethinking the principles. *Computer*, **31**, (12).

Ulieru, M., Norrie, D., Kremer, R., and Shen, W., 2000, A multi-resolution collaborative architecture for web-centric global manufacturing. *Information Sciences*, **127**, (1-2).

Vulkan, N., and Jennings, N. R., 2000, Efficient mechanisms for the supply of services in multi-agent environments. *Decision Support Systems*, **28**, (1-2).

Wu, D. J., 2001, Software agents for knowledge management: coordination in multi-agent supply chains and auctions. *Expert Systems with Applications*, **20**, (1).

Yokoo, M., Sakurai, Y., and Matsubara, S., 2001, Robust combinatorial auction protocol against false-name bids. *Artificial Intelligence*, **130**, (2).