AN ELECTRONIC MARKETPLACE ARCHITECTURE

Asuman Dogac, Ilker Durusoy, Sena Arpinar, Nesime Tatbul, and Pinar Koksal
Software Research and Development Center
Dept. of Computer Eng.
Middle East Technical University
06531, Ankara, Turkey
asuman@srdc.metu.edu.tr

ABSTRACT

In this paper, we describe a scenario for a distributed marketplace whose scope can be the whole Web where resource discovery agents find out about resources that may want to join the marketplace and electronic commerce is realised through buying agents representing the customers and the selling agents representing the resources like electronic catalogs. The marketplace contains an Intelligent Directory Service (IDS) which makes it possible for agents to find out about each other and also contains references to the related Document Type Definitions (DTDs).

We propose a possible architecture to support this scenario which is based on the emerging technologies and standards. In this architecture, the resources expose their metadata using Resource Description Framework (RDF) to be accessed by the resource discovery agents and their content through Extensible Markup Language (XML) to be accessed by the selling agents by using Document Object Model (DOM).

The IDS contains the template workflows for buying and selling agents, a trader mechanism, Resource Discovery Agents, Document Type Definitions (DTDs) and a dictionary of synonyms to be used by the buying agents to help the customer to specify the item s/he wishes to purchase. The agents and IDS communicate through KQML messages. The modifications necessary to the proposed architecture considering only the available technology are also discussed.

INTRODUCTION

Electronic commerce is a generic term that encompasses numerous information technologies and services used to implement business practices ranging from customer service to inter-corporation coordination. One of the most common instances of electronic commerce is the exchange of goods and services over the Internet. However, the electronic commerce services that are established so far are still far from being mature. There is no real integration of the available underlying technologies, and the provided services lack many important but also more challenging features.

One such feature is the automation of a marketplace on the Web through agents. For such a marketplace, there is a need for a facility which enables the semantic interoperability of resources on the Web so that buyers are able to reach the sellers that can meet their needs and vice versa. In this respect, the currently emerging standards like XML, RDF and DTDs are very promising. Furthermore, after the resources are discovered, the process of interaction between buyers and sellers (resources), that is commerce, should be automated. In other words, a virtual

marketplace on the Web should be created which not only makes buyers and sellers meet but also helps the exchange of goods between them through negotiations.

With these considerations in mind, we envision a scenario for an electronic marketplace on the Web. The distribution infrastructure of the marketplace is Web. The marketplace contains buying and selling agents as well as Intelligent Directory Service (IDS). The IDS contains the template workflows for buying and selling agents, a trader mechanism, Resource Discovery Agents, Document Type Definitions (DTDs) and a dictionary of synonyms to be used by the buying agents to help the customer to specify the item s/he wishes to purchase. The trader mechanism lets agents find out about each other.

In this scenario, resource discovery agents of IDS working in the background discover resources. If a resource is willing to join the marketplace, the IDS sends a selling agent workflow template to the seller's site. The resource creates a selling agent by automatically customizing the template by using the information obtained from the resource through a user friendly graphical interface. The resulting agent is registered with the trader. However if the resource already has a selling agent, that one is registered. The trader makes the related buying agents already in the marketplace aware of this newly created selling agent.

When a customer wants to buy a service or an item from the marketplace, s/he provides the item name to the IDS. IDS first checks the related DTDs with the item name specified by the buyer. However, the buyer may not know the right term (used in DTD) to use for the item, therefore an intelligent dictionary of synonyms is used for this purpose. For example, consider a computer shop using a computer DTD in describing its service. If a customer wants to buy a CPU and uses the term "Processor" and if "CPU" is the term used in DTD, then dictionary of synonyms is to match the word "Processor" with "CPU". Then IDS sends the names and types of the properties as well as a buying agent template and the list of related selling agents to the buyer's site. Obtaining the names and types of properties from DTDs is necessary since the buyer may not know in advance all the properties of the item. The proposed XML namespace facility can also be used together with DTD to associate DTD's and the documents.

The buying agent workflow template presents the user a form containing the values or ranges for the properties of the item along with the criteria that the customer wishes to be optimized in the negotiation phase and the required parameters. The buying agent is automatically created by customizing the workflow template using this information. The buying agent negotiates with the related selling agents to realize the transaction. A comparative analysis of the available alternatives can also be presented to the customer by the buying agent if the customer wishes so.

The paper describes an architecture for the proposed scenario, called METU-Emar, and is organized as follows: Section 2 summarizes the technologies that can be used as building blocks in the implemention. Section 3 describes the overall architecture of the marketplace. In Section 4 details of agent architecture including their implementation as workflow templates are given. Section 5 and Section 6 present the feasibility and the advantages of the proposed system. In Section 7 an alternative communication infrastructure to KQML namely CORBA is discussed. And the conclusions are given in Section 8.

RELATED TECHNOLOGIES

In this section we briefly summarize the advanced technologies and emerging standards which constitute the building blocks of the proposed architecture. In this respect, agent technology, Knowledge Query and Manipulation Language (KQML), Extensible Markup Language (XML), Document Type Definition (DTD), Resource Description Framework (RDF) and Document Object Model (DOM) are covered.

Agent Technology

Agents are programs that perform specific tasks on behalf of their users. Agents are distinguished from other types of software because they are independent entities capable of completing complex assignments without intervention, rather than as tools that must be manipulated by a user.

The fundamental properties of software agents are as follows [Woolridge, 1995]:

Autonomy: Agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state.

Social ability: Agents interact with other agents (and possibly with humans) via some kind of agent communication language.

Reactivity: Agents perceive their environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the Internet, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it.

Pro-activeness: Agents do not simply act in response to their environment, they are able to exhibit goal-directed behaviour by taking the initiative.

The agents can be made more intelligent with the following additional properties:

Rationality: Agents select actions that follow from knowledge and goals.

Adaptivity: Agents are able to modify knowledge and behaviour based on experience.

Collaboration: Agents can plan and execute multi-agent problem solving.

An earlier example of a software agent for electronic commerce is ShopBot (Shopping Robot) [Doorenbos, 1997] which is a domain-independent comparison-shopping agent. Given the home pages of several online stores, ShopBot automatically learns how to shop at these vendors. Learning process involves extracting product descriptions from home pages. This is not an easy problem because home pages may vary in format and also contain other information like advertisements and links to other sites. After learning, ShopBot is able to visit over a dozen of software vendors, extract product information, and summarize the results for the user. Preliminary results show that ShopBot enables users to both find superior prices and substantially reduce Web shopping time. ShopBot relies on a combination of heuristic search, pattern matching, and inductive learning techniques.

Yet ShopBot has several limitations. It works only on home pages that have a searchable index. It expects product descriptions to start on a fresh line. Furthermore, ShopBot heavily relies on HTML. If a vendor provides information exclusively by embedding graphics or using Java, ShopBot will be unable to handle that vendor. More importantly ShopBot shopper's performance is linear in the number of vendors it accesses which is not acceptable given the number of resources on the Web.

The Michigan Internet AuctionBot [AuctionBot] is an experimental auction server developed and operating at the University of Michigan Artificial Intelligence Laboratory. The purpose of the server is to allow anybody on the Internet to run an auction over the net. The AuctionBot provides facilities for examining ongoing auctions (which are accessible through catalog information or through keyword search), starting a new auction, biding in an existing auction and inspecting account activities. The AuctionBot provides only an information service. Registered

buyers and sellers give their bids to AuctionBot which determines a resulting allocation as entailed by a well-defined set of auction rules, and notifies the participants.

The AuctionBot simulates the auction process on a central site where the buyers and sellers meet. In other words the marketplace for the AuctionBot is a web site where buyers and sellers are registered. The sellers and buyers first access to the AuctionBot's web site through a web browser and log in to the system, then they can monitor the auctions and bid for auctions in this site.

There is extensive work on the mobile agents which have ability to transfer themselves between the systems that provide an agent server of some kind on a network. This ability lets a mobile agent move to a system that contains an object with which the agent wants to interact, then take the advantage of being in the same host or on the network as the object.

IBM's aglets [IBMAglets, 1998] are an example of a mobile agent technology. Aglet requires a host java application, an "aglet host" to be running on the destination computer. Before the aglet is transferred between the hosts, its current state is saved by using object serialization (which is available in JDK 1.1) to export the state of aglet (image of heap) to stream of bytes. When the aglet is transferred to a new aglet host, aglet host installs a security manager to enforce restrictions on the activities of untrusted aglets. Host uploads aglets through class loaders that know how to retrieve the class files and the state of an aglet from the remote aglet host. The state of the aglet is reconstructed from the stream of bytes and the aglet continues execution where it left of in the new host.

Objet Management Group is currently working on the specification of an agent framework to support agent mobility and management, named Mobile Agent Facility (MAF) [MAF] which is built on top of CORBA, thus providing the integration of traditional client/server paradigm and mobile agent technology. This work will help to ensure that different agent systems are able to work together by using standard MAFAgentSystem and MAFFinder interfaces. There are Mobile Agent Environments already developed in accordance with the MAF standards [Grasshoper, 1998].

One of the earliest examples of an electronic marketplace is Kasbah [Chavez, 1996] where users create autonomous agents that buy and sell goods on their behalf in the marketplace. Kasbah's selling agents are pro-active, they contact interested parties (namely, buying agents) and negotiate with them to find the best deal. A selling agent is autonomous in that, once released into the marketplace, it negotiates and makes decisions on its own, without requiring user intervention. Marketplace's job is to facilitate interaction between the agents by letting buying and selling agents know each other and by ensuring that they speak a common language and use a common terminology to describe the goods.

Another work that can be mentioned in this framework is OFFER which is CORBA based electronic broker, described in [Bichler, 1998]. The business model consists of suppliers, customers and electronic brokers (e-broker). Suppliers and e-brokers offer services which can be accessed over the Internet and which are procured by customers. The interfaces of these services are described in OMG's Interface Definition Language (IDL). Therefore, there is a need in establishing an interface standard on which all suppliers of a certain product category agree.

Suppliers offer an electronic catalog (e-catalog) to the customer; suppliers can also register with the e-broker. The e-broker can either maintain its own database of registered e-catalogs or it can use services of an Object Trader implemented through Trading Object Services of OMG. Hence, a customer can search for a service either directly in the catalog of a supplier or can use the e-broker to search in all the e-catalogs of all the suppliers which are registered with this broker. An

IDL interface is specified for the e-catalogs and for the e-broker which they should conform. The electronic broker described supports search in underlying catalogs and it provides a centralized marketplace with the possibility to use an auction mechanism to buy or sell goods.

Knowledge Query and Manipulation Language (KQML)

One of the requirements for software agents to interact and interoperate effectively is a common communication language (*social ability property*). KQML [Finin, 1995] is an agent communication language and a protocol developed by the Knowledge Sharing Effort (KSE) Consortium. It has been developed both as a message format and a message-handling protocol to support run-time knowledge sharing among agents which may have different content languages. It is a communication language which expresses communicative acts and it is different from the content language which expresses facts about the domain. The aim of KQML is to support computer programs in identifying, connecting with and exchanging information with other programs.

KQML language consists of three main layers: the content layer, the message layer, and the communication layer. The content layer contains the actual content of the message in the program's own representation language. This layer enables KQML to carry any message written in any representation language. The communication layer encodes a set of lower level communication parameters to the message like the identity of the sender and recipient and a unique identifier associated with the communication. The message layer is the core of KQML and determines the kinds of interactions one can have with a KQML-speaking agent. It identifies the protocol to be used to deliver the message and supplies a performative which the sender attaches to the content (such as that it is an assertion, a query, a command, or any set of known performatives). The performatives comprise a substrate on which to develop higher-level models of inter-agent interaction such as contract nets and negotiation.

In the following KQML example message John asks score-server about her score.

(ask-one:
:sender john
:receiver score-server
:content (SCORE JOHN ?score)
:reply-with c-score
:language LPROLOG)

In this message the KQML performative is ask-one, the content is (score cindy ?score), the receiver of the message is a server identified as score-server and the query is written in LPROLOG language. The value of the :content keyword is the content level, the values of :reply-with, :sender, :receiver keywords form the communication layer and the performative name (ask-one) with the :language form the message layer. In due time, score-server might send John the following message:

(tell
:sender score-server
:content (SCORE John 74)
: receiver john
:in-reply-to c-score
:language LPROLOG)

The set of performatives defined by KSE is extensible. A group of agents may agree on to use additional performatives if they agree on their interpretation and the protocol associated with

each.

The message layer also includes optional features which describe the content language, the ontology, and some type of description of the content. These features make it possible for KQML implementations to analyze, route and properly deliver messages even though their content is inaccessible.

Following are the main advantages of KQML as an agent communication language:

- KQML messages are declarative, simple, readable and extensible,
- Since KQML has a layered structure and since KQML messages are unaware of the content of the message they carry, KQML can easily be integrated with other system components,
- KQML imposes no restrictions about the transport protocol and the content language.

In addition to these, KQML has the potential to enhance the capabilities and functionality of large-scale integration and interoperability efforts in communication and information technology such as OMG's CORBA, as well as in application areas like electronic commerce [Finin, 1995].

Extensible Markup Language (XML) and Document Type Definitions (DTDs)

World Wide Web Consortium's (W3C) Extensible Markup Language (XML) [XML] defines a simple subset of SGML (the Standard Generalized Markup Language). Unlike HTML, which defines a fixed set of tags, XML allows the definition of customized markup languages with application specific tags [Manola, 1998]. That is, XML provides support for the representation of data in terms of attribute/value pairs with user defined tags.

XML differs from HTML in three major respects [Bosak, 1998]:

1. Information providers can define new tag and attribute names at will.
2. Document structures can be nested to any level of complexity.
3. Any XML document can contain an optional description of its grammar for use by applications that need to perform structural validation.

Document Type Definitions (DTD) which are defined for user groups provide a formal definition of documents for that group, that is, they define what names can be used for elements, where they may occur and how they all fit together in an XML file.

The XML working group is currently developing a facility [XMLNames] that will allow tag names to have a prefix which make them unique and prevent name clashes when developing documents that mix elements from different schemas. XML namespace proposal provides a way to define relationship between the use of particular element name in a document and the standard vocabulary (DTD) from which the name is taken.

As an example, the following XML file (staff.xml)can be used to represent the staff report for a university. The related DTD states that faculties have a NAME attribute and contain department elements. Departments have a NAME attribute and contain a STAFF and a STUD elements. STAFF shows the number of full-time and part-time staff, while STUD gives the number of graduate and undergraduate students.

Staff.xml :

```
<?XML VERSION="1.0" ?>
<!DOCTYPE STAFFREPORT "staff.dtd">
<STAFFREPORT>
<FACULTY NAME="ENG">
<DEPARTMENT NAME="CS">
<STAFF FT="30" PT="10"/>
<STUD G="60" U="300"/>
</DEPARTMENT>
<DEPARTMENT NAME="EE">
<STAFF FT="55" PT="10"/>
<STUD G="70" U="600"/>
</DEPARTMENT>
</FACULTY>
</STAFFREPORT>
```

Staff.dtd

```
<!ELEMENT FACULTY (DEPARTMENT)+>
<!ATTLIST FACULTY NAME CDATA #REQUIRED">

<!ELEMENT DEPARTMENT (STAFF, STUD)>
<!ATTLIST DEPARTMENT NAME CDATA #REQUIRED">

<!ELEMENT STAFF EMPTY>
<!ATTLIST STAFF NAME FT CDATA #REQUIRED">
<!ATTLIST STAFF NAME PT CDATA #REQUIRED">

<!ELEMENT STUD EMPTY>
<!ATTLIST STUD NAME GS CDATA #REQUIRED">
<!ATTLIST STUD NAME US CDATA #REQUIRED">
```

Resource Description Framework (RDF)

As ShopBot's limitations given in Section 2.1 clearly demonstrated, there is a need for machine understandable information on the Web. An emerging solution to letting automated agents surf the Web is to provide a mechanism which allows a more precise description of the resources that are available on the Web [Lassila, 1998a]. The Resource Description Framework (RDF) [RDF] by the World Wide Web Consortium (W3C) is a standard for metadata that provides interoperability between applications that exchange machine-understandable information on the Web.

RDF [RDF] defines both a data model for representing RDF metadata, and an XML-based syntax for expressing and transporting metadata. RDF is a model for representing named properties and their values. These properties serve both to represent attributes of resources and to represent relationship between resources. The RDF data model is syntax independent way of representing RDF expressions and in [RDF], three representations of the model are given, that is, representation as 3-tuples, as a graph and in XML.

In 3-tuple representation, a property is a three tuple consisting of the resource being described, a property name or type, and a value. A collection of property triples describing the same resource is called an *assertion*. In graph representation, the resources being described and the values describing them are nodes in a directed graph, with the edges being labelled by the property

names. An RDF statement can itself be the target node of an arc (i.e., the value of some property) or the source node of an arc (i.e., it can have properties). In these cases, the original property (i.e., the statement) must be reified; that is, converted into nodes and arcs. Reified properties are drawn as a single node with several arcs emanating from it representing the resource, property name and value [Manola].

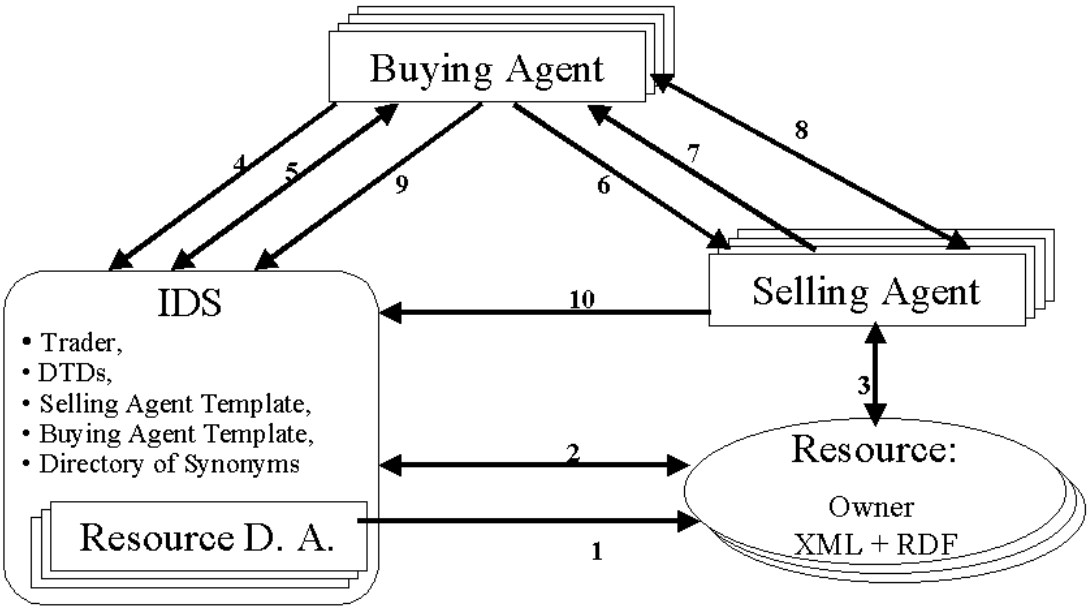
It is clear that RDF will provide the much needed information for the agent technologies working on the Web. Agents can use RDF not only for describing their capabilities and negotiating the terminologies used in communication, but also the other resources on the Web.

The following example gives the meta-data about the document resource (http:://www.x.com/y.xml). The author of the resource is '|ohn Doe', whose name is "John Doe" and title is "manager". The namespace prefixes are used to show that the author property is taken from the schema defined in /XML/sbib.

```
<?xml:namspace name="/XML/sbib" as="bib"?>
<?xml:namspace name="http://../rdf-s" as="RDF"?>
<RDF:serialization>
<RDF:assertion href="http:://www.x.com/y.xml">
<bib:author>
<RDF:resource>
<bib:name>John Doe</bib:name>
<bib:title>manager</bib:title>
</RDF:resource>
</bib:author>
</RDF:assertion>
</RDF:serialization>
```

Document Object Model (DOM)

W3C's Document Object Model (DOM) [DOM, 1998] defines an object-oriented API for HTML and XML documents which a Web client can present to programs that need to process the documents [Manola, 1998]. DOM represents a document as a hierarchy of objects with proper inheritance relationship among them, called nodes, which are derived (by parsing) from a source representation of a document (HTML or XML). In other words, the DOM object classes represent generic components of a document, and hence define a document object meta model. The major DOM classes are: Node, Document, Element, Attribute, Text, Processing Instruction, and Comment. The representation of a Web page in terms of objects makes it easy to associate code with the various subcomponents of the page. For example, Document object has a "documentType" method which returns DTD for XML documents (and "null" for HTML and XML documents without DTDs) and a "getElementsByTagName" method which produces an enumerator that iterates over all Element nodes within the document whose "tagName" matches the input name provided. Thus DOM provides a general means for applications to access and traverse documents written in HTML and XML without having themselves to perform complex parsing.



Phase I: 1. RDA find out about Resources and informs owners about METU-EMar.
2. Resource loads SA template, then configures SA and registers with the trader.
3. SA accesses Resource through DOM.

Phase II: 4. Buyer contacts with IDS, and gives the name of the item.
5. Buyer loads BA template, item properties and a list of SAs, then creates and registers BA with the trader.

Phase III: 6. BA asks related SAs about the required item and requests proposals.
7. SAs respond with their proposals if they sell the required item (Alternatively send negative response).

Phase IV: 8. Parties perform negotiation.

Phase V: 9. BA informs IDS about the completion of the process. IDS removes it from the trader.
10.SA informs IDS if it is no longer selling the item. IDS remove it from the trader.

Figure 1: The Architecture of METU-Emar

METU-EMAR ARCHITECTURE

A possible architecture realizing the scenario given in Section 1 that uses the technology summarized in Section 2 is described in the following (Figure 1):

The electronic commerce in the marketplace is realized in five phases. Resource Discovery Agents finds out resources on the Web and the selling agents are created in Phase I. In Phase II a buying agent is created for the customer that operates in the marketplace in order to make a purchase. In Phase III, the buying agents contact with the all possible selling agents to determine the ones that can provide the item with required properties. Phase IV contains the negotiation between buying and selling agents. Finally, the result of the transaction is reported to the IDS to make proper updates like deleting the agent that leaves the marketplace from the trader.

PHASE I:

The resource discovery agents working in the background find out about the resources providing products and services. We expect resources to expose their semantics by using the Resource Description Framework (RDF) [Lassila, 1998b] and the Extensible Markup Language (XML) [XML]. As briefly summarized in Section 2.4, RDF defines both a data model for representing RDF metadata, and an XML-based syntax for expressing and transporting the metadata. Since resources use RDF to expose their metadata, the resource discovery agents do not need

intelligence in extracting information from the resources. However, they do have other properties of agents like being autonomous, reactive and proactive.

Resource Discovery Agents contact the owner of the resource by using the means (possibly an e-mail address) available on the resource. If the resource wants to join the marketplace, the owner of the resource accesses to the marketplace's web site through a web browser and marketplace provides it with a template workflow of a selling agent.

Then the owner of the resource (user) configure the workflow agent template by filling the form generated by the template. The form contains information automaticaly extracted from the resource (items or services provided, their prices) and request information from the user (owner of the resource) about the negotiation policy and the desired user interaction characteristics like how and when the user wants to be informed about the progress of the negotiation. On complition of the form, the selling agent is automatically created and registered with the trader of the IDS. If the resource already has a selling agent, this one is registered.

PHASE II:

When a customer wants to make a purchase on the marketplace, s/he connects to the marketplace's web site through a web browser and specifies a service or a product s/he wishes to purchase from the marketplace. The IDS that contains references to the DTDs, accesses to the related Document Type Definitions (DTDs) to obtain names and types of properties of the product. DTDs which are defined for customer groups provide a formal definition of documents for that group, that is, DTDs define what names can be used for elements, where they may occur and how they all fit together in an XML file as described in Section 2.3. In our case, all the merchants use the same definition in their DTDs for the item accessed by the selling agent. Therefore, there is no need for a translation among terminologies (which is necessary when XML files have different DTDs and different customers define their own ways of using attribute/value pairs to represent the same information).

Different names can be provided for the same product by the customers, in other words, the customers may not know the standard terms used in DTDs. Therefore, a dictionary of synonyms is necessary in the IDS. This dictionary of synonyms may be implemented to contain some intelligence in the sense that whenever an item or service is not found in the dictionary, the customer may be asked to provide synonyms and these terms can be added to the dictionary for later use.

Then IDS sends a buying agent workflow template, the names and types of the properties of the item and a list of related selling agents to the buyer's site. The customer (user) using the standard forms that is presented to him specifies the desired property values of the item, and the negotiation strategy. Using this information the buying agent is automatically created and registered with the trader.

In filling out the form, the buyer defines all the required properties of the item. Some of those properties may be open to negotiation while the others may not be negotiable. For example, the car item has a property "year", which represent the year in which the car is produced. If the property is determined as non-negotiable the selling agents are not allowed to give a proposal that may have a different value than the required one.

PHASE III:

A buying agent contacts all the related selling agents using the list provided by the IDS in Phase II and asks them if they can provide the item with the desired properties and conditions. For

example the buying agent which wants to buy a "second hand domestic car" has the list of all the selling agents that sell cars. But only some of them may have the car with desired properties ("second hand" and "domestic").

All the sellers respond to the request of the buyer with the list of proposals for matching items that they sell or send a negative response if their items do not meet the required properties. The proposals contain all the negotiable properties that the sellers offer including the price.

Using the responses from the selling agents and considering the hints and restrictions given by the user, buying agent determines the buying strategy. For example, if a selling agent with a bargaining facility is already giving a lower price than a selling agent without a negotiation facility, the second is eliminated. Such a strategy is also possible for the selling agents. In other words, the buying and selling agents are playing a game where each is trying to satisfy its goals. The buying agents are on the customers' side and the selling agents are on the resources' side.

PHASE IV:

The buying agents go in a direct negotiation with selling agents by sending and receiving proposals. In this respect, RDF is used in encoding resources and query capabilities and KQML [Labrou, 1997] is used to communicate RDF among agents.

The buying and selling agents in the marketplace act autonomously, that is, once released in the marketplace, they negotiate and make decisions on their own, without requiring customer intervention. They are proactive in contacting the other interested agents and reactive to the changes in the marketplace like new agents.

The resources should provide semantic information about their content to the selling agent. In this respect, the resources should be defined in XML. DOM is used by the selling agents in processing XML pages to obtain specific product data, like the price of the product. The selling agents should be authorized to invoke certain applications at the resource to obtain the bargaining strategy and its parameters which implies that the resources should provide this information through a standard interface.

The negotiation strategies as described in [Chavez, 1996] can be used in the negotiation phase. Several parameters can be specified, like the desired date to sell (buy) the item, desired price, lowest (highest) acceptable price and a decay function if the agent wants to decrease (increase) the price over its given time frame. However there is a need for more solid bargaining algorithms [Bichler, 1998].

PHASE V:

This phase starts after the successful completion of a negotiation. When the negotiation is complete, the buying agent informs the IDS that it is done. Then the IDS removes the buying agent from the trader. Also in completion of the negotiation, all the items of the selling agent may be sold. In this case the selling agent informs IDS about it and IDS updates the trader.

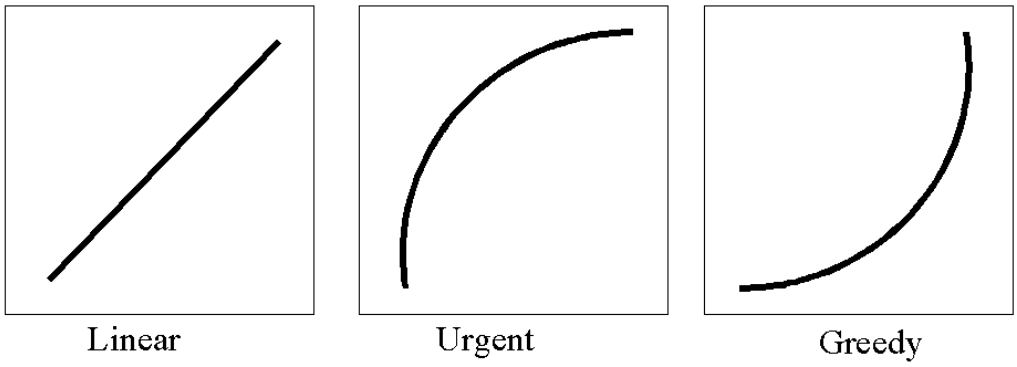


Figure 2: The Price Function for Buying Agent

Initial Configuration of an Agent

The user (buyer or seller) configures the agent template by filling in the form provided by the template. The form includes the questions about Property Parameters, Price Function, User Interaction Characteristics and User Preferences in the negotiation.

Property parameters are used to evaluate a proposal. To be able to compare the received proposals, it is necessary to evaluate a proposal to a numeric value. For this purpose negotiable properties are assigned with two parameters. First each property has a user defined weight (over 100) which reflects the comparative importance of the property among other negotiable properties. Second for the properties whose values lie within a desired range, a function is evaluated that returns a real number between 0 and 1. By multiplying the weight and the result of the function, the value of the property is obtained. By summing up all the property values, the value of the proposal is obtained.

As an example, assume that a user wants to buy a "domestic second hand automobile" and defines the property "year" as negotiable. First he is asked the weight of the year property. Second the user is asked to give the values for at least two years and a function type (linear or exponential). Then using a curve fitting algorithm, the function is obtained for the year. For example, user may give 0.8 for the year 1998, 0.4 for the year 1994, and may define function type as linear. In this case, the function becomes (0.1*x - 199) and the year 1996's value becomes 0.6 (The important point in the function is that: it allways returns a value between 0 and 1)/

Price Function is used by the agent to define the buying or selling prices during negotiations. It represents the rate at which agents reduces (selling agent) or increases (buyer agent) the price. As it is in KASBAH [Chavez, 1996] 3 types of function can be selected (Figure 2).

The User Interaction Characteristics defines how and when the user is to be informed or his interaction is required in negotiation. User selects either e-mail or message-box as a vehicle for being informed. The possible alternatives for informing the user include:

- inform (and wait) at every step,
- inform (and wait) at the last step,
- inform (and wait) when price exceeds a certain value,

- inform (and wait) when time exceeds a certain value.

AGENT ARCHITECTURE

There are three types of agents in the system namely Resource Discovery Agents, Buying and Selling Agents. The resource Discovery Agents work in the background and find out about resources that may want to join the marketplace. The Buying and Selling on the other hand perform automated negotiation based on a strategy hinted by the user. The architecture of the Resource Discovery Agents is simpler compared to the Buying and Selling agents.

Resource Discovery Agents

The Resource Discovery Agents of the IDS are as the name implies responsible for finding the XML resources on the whole Web. This functionality is achieved by searching all the web sites and following the links to find related resources. Since the resources that are acceptable to the marketplace present the metadata about themselves in RDF, the job of Resource Discovery Agents is not complex.

Once Resource Discovery Agents obtain an XML page, the RDF description is accessed and metadata available in the RDF is processed in order to understand the content of the XML document (the resource). If the resource provides services or sells items, then the Resource Discovery Agent contacts the owner of the resource.

Information about the contact points are obtained from the RDF metadata, which gives all the information about the resource including the owner, the author and the mail address. A possible contact mechanism is e-mail. The Resource Discovery Agent sends an e-mail to the owner of the resource describing the facilities of the Marketplace and the Web address of the marketplace.

Then it is the resource owner's choice to join the marketplace. If s/he decides to join the marketplace s/he can connect to Marketplace's web site and can download the selling agent template for customization.

In summary, Resource Discovery Agents are simple programs which do not require to be re-configured or dynamically modified at run time.

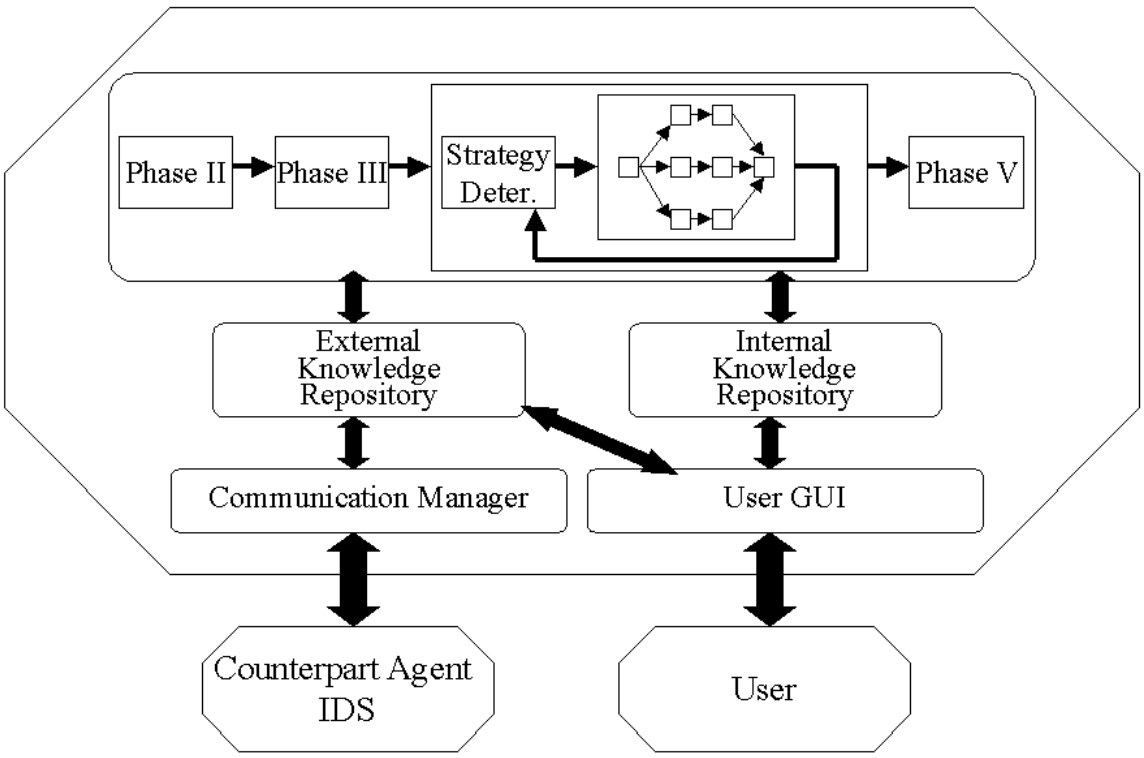


Figure 3: The Architecture of Agents

Buying and Selling Agents

In our system, the templates of buying and selling agents exist in the IDS and are provided to buyers and sellers to automatically reconfigure their agents. The agents deploy workflow technology that provides for easy and controlled re-structuring and dynamic modification facilities which are critical for customization and adaptation of agents.

Workflow approach allows buyers and sellers to define their own tasks and to invoke their already existing applications within the workflow and to re-structure the control and data flow between the tasks, in other words, to automatically create a custom built workflow from the workflow template. The higher level of abstraction provided by the workflow technology makes this customization of agents for different users possible.

Our agent architecture necessitates the customization of workflow both at compile time and at run time. At compile time, according to the specifications in the forms, the particular activities to be invoked and their control flow is decided. For example the user may not wish to use an automated payment process in which case the payment activity is not included in the workflow specification generated for that customer.

Once the agent (workflow) structure is defined by the user the agent can be activated at the user's site. The agents perform straightforward tasks during the Phases I,II,III and V of the interaction

protocol. But the tasks in the Phase IV which involves the negotiation between the agents can not be defined from the beginning. Instead they are created and organized dynamically depending on:

(1) the responses/proposals from the counterpart agents,

(2) the interaction characteristics of the user and

(3) the outcomes of the strategic reasoning.

Therefore the structure and control flow between the tasks in Phase IV must be defined dynamically and should be modified on the fly. These features require workflow technology which allow for run time modifications.

The agents at every step evaluate the inputs received from counterpart agents and users and determine a negotiation strategy. Then agents create new components to the workflow which are responsible for realizing this strategy. An example to the changes that may be necessary at run time is the following: As a result of the buying strategy, different patterns of invocations may be produced. For example, buying strategy may decide to contact two selling agents with very similar offers in parallel, or it may choose to contact the most promising selling agent first and try the others only if it can not succeed with this one.

The added advantage of using workflow technology in realizing agents is the recovery facility provided. As an example a selling agent may have committed a payment activity after which the shipment activity fails. In this case the selling agent can be automatically rolled back by the workflow system by executing a compensation activity for the payment activity.

The workflow scheduler of the agents must have a distributed scheduler since a centralized scheduler is a potential bottleneck.

Workflow Agent Implementation

We have developed a workflow system architecture to support the agent technology described. The workflow has componentized architecture and uses the block structured specification language described in [Dogac, 1998a]. The various types of blocks defined in the specification language constitute the components of the workflow. Blocks can be nested and the top level block represents the workflow process. Since the block structured language confines intertask dependencies to a well formed structure, the automated generation of agent from a workflow template becomes possible.

When it comes to the scheduling policy of the workflow, the described agent architecture seems to have somewhat conflicting requirements. The scheduling should be distributed to avoid centalized scheduler bottleneck where the whole agent workflow comes to a halt when the centralized scheduling site fails. In distributed scheduling there are several schedulers running on different nodes of the network each executing a part of a process instance so that the execution can continue even when some of the sites fail. As an example if the site where the negotiation is handled fails after sending the appropriate messages to the rest of the sites, the workflow agent can successfully keep on scheduling and executing the remaining parts of workflow like payment or shipment.

Yet when it comes to run time modification of a workflow with a distributed scheduling mechanism, finding out about the pieces of a process instance and exerting control on them becomes very inefficient.

Our solution to this problem is as follows: We handle the run time modifications within the scope of a block which nicely fits with the requirements of agents. We exert central control over the activities in the block with a block scheduler which determines the execution sequence of activities within its scope. Yet the sheduling of the blocks within a workflow are distributed.

Hence having central schedulers for blocks facilitate run time modifications on the blocks, and having blocks scheduled in a distributed manner increases the failure resiliency and performance of the system.

Components of Agents

The buying and selling agents in our architecture contain some standard modules as indicated in Figure 3, which are:

1. Communication Manager,
2. Internal Knowledge Repository,
3. External Knowledge Repository, and
4. User GUI.

In our model agents communicate with each other and with IDS directly by sending and receiving KQML messages. The Communication Manager is responsible for receiving and sending these messages in a convenient and secure way. It contains low level information related to communication and security and provides abstraction to the other modules. It is the communication manager that

- obtains the network addresses of the counter parties from IDS and stores them locally to prevent communication overhead and maps agent ids to the network addresses,
- performs message format conversion that gets KQML messages from the other agents and converts them to internal structure or that gets messages from higher levels and converts them to associated KQML messages
- ensures security (if requested) by authentication, and encryption and decryption of the messages. In this respect the trader can give a pair of keys (public and private) to the agents when they are initially created to be used in the negotiation phase. When the agents are informed about the counter agents they are provided with the public keys of these agents which they can use to encrypt messages to be sent to them and to perform authentication.

It should be noted that when the Communication manager is implemented as a separate component the interagent communication method can be easily modified to CORBA or email by modifiying this component.
The Internal Knowledge Repository stores and provides information about the internal structure of the agent itself. This information contains

- the product information which are names, range, desired values and weights for properties and
- strategy information that is used in determining the negotiation strategy, which includes deadlines, price determination function and preferences of the user

Both selling and buying agents keeps the state information to be used in determining the negotiation strategy. The External Knowledge Repository stores and provides information about

the counterpart agents. This information includes:

- the counterpart agent information such as id, location etc.
- history of the negotiation for each agent which is exploited in determining negotiation strategy and
- current state for each negotiated item, that is current bid and their evaluated value.

User interaction is critical for the success of the agents. The user GUI is responsible for the interaction between the agent and the user. This interaction involves:

- initial configuration of the workflow,
- monitoring of the progress by the user and
- direct intervention of the user to the negotiation strategy determination activity.

An Alternative: Mobile Agents

An alternative to realize agents is to use mobile agent paradigm. It is obvious that mobile agents can make certain applications easier to develop, or improve reliability and efficiency when used in the proper content. For example in applications that depend on collaboration among a large number of people who are not colocated, people can be represented by mobile agents, gathering in one central location to do their collaboration and then returning home to display details. Another area in which mobile agents may be useful is in processing large amounts of data over low-reliability networks where mobile agents are transferred to the locations that data resides to perform processing and are returned to home to present results [Kinnery, 1998]

In spite of the stated advantages of the mobile agents, they have some drawbacks. Firstly in order for them to visit a site, the site must run a special agent server software, hence it is not possible to move agents to every site. Secondly, security is a very important issue to consider: when a mobile agent which contains private information moves to another host, one should be certain that its information will not be broken. In our scenario we prefer to use stationary agents rather than mobile agents for the following reasons:

1. Our agents do not perform large amount data processing. Therefore it is not critical to take advantage of locality during the operations.
2. The selling and buying agents are created by using templates on the sites of buyer or seller. It may not be possible or desirable to load and run the mobile agent host systems on these sites.
3. The buying agents may contain private information such as credit card number, and the owners may not want it to be transferred to another system over the network.
4. The buying and selling agents may require (depending on the desires of the owner) owner interaction which makes it desirable to run the agents on the site of the owners so that they can easily communicate and monitor through GUI.

FEASIBILITY

The technological requirement of the architecture proposed is the semantic interoperability of the Web resources. The building blocks for this, although have already been defined or are being defined mostly as standards, are at their infancy. For example, work is underway to define XML-based data exchange formats in both the chemical and the health care communities. A number of

industry groups defined SGML DTDs for their documents (e.g. the US Defense Department, which requires much of its documentation to be submitted according to SGML DTDs)[Manola, 1998]. A large US project aims to define specific property names for specific elements in computer industry that can possibly be implemented through XML DTDs [Danish, 1998].

The architecture we describe requires the DTDs for the user groups to be available. Note that since RDF assertions use properties defined in the schemas, i.e., DTDs, the use of RDF also depends on the availability of standard DTDs. Until the standard DTDs become available and the RDFs start using these schemas, there is a need for the following modifications in METU-EMar architecture in realizing the proposed scenario:

1. The resource discovery agents utilize machine understandable information (RDF) and therefore can not be implemented easily when the standard vocabulary (DTDs) used by RDF is not available. In this case, resource discovery agents should either be more intelligent or include heuristic techniques to understand the content of the resources.

2. When XML files have different DTDs (i.e., different users define their own ways of using attribute/value pairs to represent the same information), there is a need for a mechanism to identify associations among the terminologies of the XML files. This can be achieved through a translation mechanism between terminologies. This translation is also needed in the negotiation phase among the buying and the selling agents.

Also as stated previously, more solid bargaining algorithms must be developed [Bichler, 1998] to better exploit the scheme described.

ADVANTAGES

It is clear that in a marketplace as large as the one provided by the Web, the service provided by the proposed architecture is invaluable. It will not only help to locate better opportunities for both the buyers and the sellers but it will also save a lot of their time in negotiations. In other words, the proposed marketplace aims to find the best conditions for its clients and help to overcome the limitations of direct communications between customers and suppliers. The marketplace enables the customers to reach various suppliers whose existence they are unaware of and hence it would be impossible for them to reach otherwise. Symmetrically, the marketplace also gives the suppliers the chance to contact to a much wider range of customers.

AN ALTERNATIVE APPROACH TO KQML: CORBA

CORBA together with the Web can also be used as the distribution infrastructure. In this case, all the agents in the system and IDS can be implemented as CORBA objects. Using CORBA as the infrastructure provides the opportunity to use OMG's Trading Object Service as a trader of IDS.

CORBA as Distribution Infrastructure

Web itself and the distributed object platforms like CORBA or Active X/DCOM provide a distribution infrastructure. It is possible to use the Web (HTTP, HTML and Java) in conjunction with an object-oriented "communication bus" following Common Object Request Broker Architecture's (CORBA) Object Request Brokers (CORBA 2.0 and IIOP). Indeed, these sets of technologies constitute the basis of some of the major electronic commerce platforms like Netscape ONE (Open Network Environment), Oracle's NCA (Network Computing Architecture), IBM's CommercePoint and Sun and JavaSoft's Java Electronic Commerce Framework [Tanenbaum, 1997].

Using CORBA 2.0 and IIOP with Web rather than using Web alone provides the following advantages [Orfali, 1997]:

1. In Web, method invocation is realized through HTTP and Common Gateway Interface (CGI) protocol. When this HTTP/CGI layer is replaced by CORBA, since CORBA allows clients to directly invoke methods on a server, a lot of overhead is avoided. Furthermore, any IDL defined method on the server can be invoked and typed parameters can be passed instead of just strings.

2. With CGI, a new instance of a program must be started every time an applet invokes a method on the server. With CORBA, the same server object receives successive calls from the client and preserves the state between these invocations.

3. CGI is a stateless protocol, that is, CGI does not maintain information from one form to the next. Therefore, hidden fields within a form are used to maintain state on the client side. CORBA maintains the state between client invocations avoiding this overhead, too.

4. CGI creates a bottleneck because it has no way to distribute the incoming requests across multiple processes and processors. CORBA ORBs on the other hand can create as many server objects as necessary. These server objects can run on multiple servers to provide load balancing for incoming client requests.

5. With CORBA, Java clients and applets can invoke a wide variety of IDL defined operations on the server. In contrast, HTTP clients are restricted to a limited set of operations.

6. CORBA provides a rich set of distributed object services that augment Java, including trader, transactions, security, naming and persistence.

It should be noted that, like HTTP, CORBA's IIOP uses Internet as the backbone. This means that both IIOP and HTTP can run on the same networks.

As a summary, CORBA in conjunction with Web seems to be a very promising infrastructure for electronic commerce applications.

Trading Object Service (TOS) as Trader

The OMG Trading Object Service [OMG, 1996] facilitates the offering and the discovery of instances of services of particular types. A trader is an object that supports the Trading Object Service in a distributed environment. It can be viewed as an object through which other objects can advertise their capabilities and match their needs against advertised capabilities. Advertising a capability or offering a service is called "export". Matching against needs or discovering services is called "import". Export and import facilitate dynamic discovery of, and late binding to, services.

To export, an object gives the trader a description of a service together with the location of an interface at which that service is available. To import, an object asks the trader for a service having certain characteristics. The trader checks against the service descriptions it holds and responds the importer with the location of the selected service's interface. The importer is then able to interact with the service.

Due to the sheer number of service offers that will be available worldwide, and the differing requirements that users of a trading service will have, it is inevitable that a trading service will be split up and that the service offers will be partitioned. Traders in different partitions interact with each other to answer the needs of a client.

Comparison of KQML with CORBA in Communication

CORBA provides the following basic advantages to our architecture:

- CORBA provides simpler communication mechanism among the agents and the IDS, compared to KQML which introduces a lot of communication overhead. When the agents and the IDS are defined as CORBA objects, the communication among them can be handled using predefined CORBA interfaces rather than the KQML messages. When communication among the objects is required the objects (agents or IDS) invoke each other's predefined methods using IIOP and passing the parameters instead of sending KQML messages.

- Using CORBA as the infrastructure provides the opportunity to use OMG's Trading Object Service as a trader of IDS. The selling agents of the resources as well as the buying agents can be registered to the related trader objects through the "Register" interface of this service and the buying agents find out about the selling agents through the "Lookup" interface and vice versa. Trading Object Service is distributed in the sense that several traders can be linked through the "Link" interface and can be searched depending on the prespecified policies.

Using CORBA as the infrastructure, on the other hand, requires an ORB at every site of the marketplace which may restrict the scope of the marketplace. Also it may be necessary to provide an ORB specific worklow template.

CONCLUSIONS

The Internet is revolutionizing commerce. However, closed markets that cannot use each other's services, incompatible applications and frameworks that can not interoperate or build upon each other are hampering the progress of electronic commerce [Tanenbaum, 1997].

The need for semantic interoperability of the resources on the Web resulted in a series of standardization efforts from the World Wide Web Consortium. In this paper, we present an electronic market that exploits these standards as well as some other emerging technologies like workflow agents. We also present a workflow agent architecture to meet the demands of buying and selling agents in the marketplace presented.

REFERENCES

(AuctionBot) http://auction.eecs.umich.edu.

(Bichler, 1998) Bichler, M., Beam, C., Segev, A., "Offer: A Broker-centered Object Framework for Electronic Requisitioning", in Proc. of Intl. IFIP Working Conference: Trends in Electronic Commerce, Hamburg, Germany, June 1998.

(Bosak, 1998) Bosak, J., "XML, Java, and the Future of the Web", http://sunsite.unc.edu/pub/sun-info/standards/xml/why /xmlapps.html.

(Chavez, 1996) Chavez, A., Maes, P., "Kasbah: An Agent Marketplace for Buying and Selling Goods", Proc. of the First Intl. Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, London, UK, April 1996, http://agents.www.media.mit.edu:80/groups/agents/Publications/kasbah-paam96.ps.

(Danish, 1998) Danish, S., Personal Communication.

(Dogac, 1998a) Dogac, A., Gokkoca, E., Arpinar, S., Koksal, P., Cingil, I., Arpinar, B., Tatbul, N., Karagoz, P., Halici, U., Altinel, M., "Design and Implementation of a Distributed Workflow Management System: METUFlow", in (Dogac, 1998b).

(Dogac, 1998b) Dogac, A., Kalinichenko, L., Ozsu, T., Sheth, A., (Edtrs.), "Advances in Workflow Management Systems and Interoperability", Springer-Verlag, 1998.

(DOM, 1998) Document Object Model (DOM), http://www.w3.org/DOM/.

(Doorenbos, 1997) Doorenbos, R. B., Etzioni, O., Weld, D. S., "A Scalable Comparison-Shopping Agent for the World- Wide Web", ACM Agents '97 Conference, 1997.

(Finin, 1995) Finin, T., Labrou, Y., Mayfield, J., "KQML as an agent communication language", in Jeffery M. Bradshaw, editor, Software Agents, MIT Press, 1995.

(Grasshoper, 1998) http://www.ikv.de/products/grasshoper/ architecture.html.

(IBMAglets, 1998) http://www.trl.ibm.com/aglets/

(Kinnery, 1997) J. Kinnery, D. Zimmerman "A hands-on look at java mobile agents " IEEE Internet Computing, August 1997.

(Koksal, 1998) Koksal, P., Arpinar, S., Dogac, A., "Workflow History Management", ACM Sigmod Record, Vol. 27, No. 1, March 1998.

(Labrou, 1997) Labrou, Y., Finin, T., "A Proposal for a new KQML Specification", Report TR-97-03, Computer Science and Electrical Engineering Department, University of Maryland Baltimore County. Available on-line as http://www.cs.umbc.edu/jklabrou/publications/tr9703.ps.

(Lassila, 1998a) Lassila, O., "RDF Metadata and Agent Architectures", http://www.objs.com/ workshops/ws9801/papers/paper056.html.

(Lassila, 1998b)Lassila, O., Swick, R. R. "Resource Description Framework (RDF) Model and Syntax", Working Draft, World Wide Web Consortium. Available on-line as http://www.w3.org/TR/WD-rdf-syntax/.

(MAF, 1998) Mobile Agent Facility, http://www.genmagic.com./agents/MAF.

(Manola, 1998a) Manola, F., "Towards a Web Object Model", http://www.objs. com/OSA/wom.htm.

(Manola, 1998b) Manola, F., "Towards a Richer Web Object Model", ACM Sigmod Record, Vol. 27, No. 1, March 1998.

(Muth, 1998) Muth, P., Weissenfels, J., Weikum, G., "What Workflow Technology Can Do for Electronic Commerce", in Current Trends in Database Technology, Dogac, A., Khosrowpour, M., Ozsu, T., Ulusoy, O., (Edtrs.), Idea Group Publishing, 1998.

(Orfali, 1997) Orfali, R., Harkey, D., "The Essential Client/Server Programming with JAVA and CORBA", John Wiley, 1997.

(OMG, 1996) OMG's Trading Object Service. OMG Document orbos/96-05-06, Version 1.0.0, May 10, 1996.

(Tanenbaum, 1997) Tanenbaum, J. M., "Eco System: An Internet Commerce Architecture", IEEE Computer, Vol. 30, No. 5, May 1997.

(RDF, 1998) Resource Description Framework (RDF), http://www.w3.org/Metadata/RDF/.

(Woolridge, 1995) Woolridge, M., Jennings, N, "Intelligent Agents- Theory and Practice", Knowledge Engineering Journal, June 1995.

(XML, 1998) Extensible Markup Language (XML), http://www.w3.org/XML/.

(XMLNames, 1998) http://www.w3.org/TR/1998/NOTE-xml-names-0119.