METU-EMar: An Agent-Based Electronic Marketplace on the Web *

Asuman Dogac, Ilker Durusoy, Sena Arpinar, Esin Gokkoca, 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 on the Web where resource discovery agents find out about resources that may want to join the marketplace and electronic commerce is realized through buying agents representing the customers and the selling agents representing the resources like electronic catalogs.
We propose a possible architecture 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 marketplace contains 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. Distribution infrastructure is CORBA and Web on which the buying and selling agents find out about each other using Trading Object Services. The modifications necessary to the proposed architecture considering only the available technology are also discussed.

1 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.

* This work is partially being supported by Middle East Technical University, the Graduate School of Natural and Applied Sciences, Project Number: AFP-97-07.02.08 and by the Scientific and Technical Research Council of Turkey, Project Number: 197E038

C. Nikolaou, C. Stephanidis (Eds.): ECDL'98, LNCS 1513, pp. 777–790, 1998.

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 RDF and DTDs sound 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. Intelligent software agent and workflow technologies are the means through which this automation can be achieved.

With these considerations in mind, we envision a scenario for an electronic marketplace on the Web. The distribution infrastructure of the marketplace is CORBA and Web. The marketplace, in addition to resource discovery agents, contains templates of buying and selling agents, pointers to the Document Type Definitions (DTDs), trader objects implemented through Trading Object Service and an intelligent dictionary of synonyms.

In this scenario, resource discovery agents working in the background discover resources. If a resource is willing to join the marketplace, the marketplace creates a selling agent workflow template for the resource and registers it with the trader. However if the resource already has a selling agent, that one is registered. If there are any related buying agents already in the marketplace when the selling agent is registered, the trader makes the selling agent aware of this buying agent.

When a customer specifies a service or an item s/he wishes to purchase from the marketplace, a buying agent workflow template is created for the customer. The customer 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". The buying agent contacts the marketplace and obtains a form for the customer to specify the properties of the item s/he wants to buy which contains the names and types of the properties of the item. Such a form which is created using the information in the related DTD, is necessary since the customer may not know in advance all the properties of the item.

The buying agent gets the filled form containing the values or ranges for the properties of the item from the customer along with the criteria that s/he wishes to be optimized in the negotiation phase and the required parameters. 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 rest of this paper describes the related technologies and an architecture for realizing this scenario. Section 2 summarizes the technologies that can be used as building blocks in implementing the proposed scenario. Section 3 describes

the architecture and discusses its feasibility and the advantages. In Section 4 related work is presented and the conclusions are given in Section 5.

2 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, current distribution infrastructures, Trading Object Service, agent technology, workflow agents, Knowledge Query and Manipulation Language (KQML), Resource Description Framework (RDF), Extensible Markup Language (XML), Document Type Definition (DTD), and Document Object Model (DOM) are covered.

2.1 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 [17].

Using CORBA 2.0 and IIOP with Web rather than using Web alone provides the following advantages [16]:

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.

2.2 Trading Object Service

The OMG Trading Object Service [20] 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.

2.3 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 [22]:

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 behavior 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 behavior 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 [7] 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.

2.4 Workflow Agents

Coupling agent technology with workflow systems seems to be a very promising research direction since these technologies nicely complement each other. The resultant "intelligent workflow" will not only have the properties of intelligent agents like being reactive, intelligent and adaptive but also will define an agent consisting of processing steps with data and control flow among them. However, to be used in agent construction, the current workflow technology must be improved in several directions including better support for ad hoc workflows and modifications at run time as well as truly distributed enactment service [5,6].

2.5 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 [8] 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. 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 [8].

2.6 Resource Description Framework (RDF)

As ShopBot's limitations given in Section 2.3 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 [11]. The Resource Description Framework (RDF) [21] 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 [21] 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 [21], 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 *assertions*. 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 [13].

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.

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

World Wide Web Consortium's (W3C) Extensible Markup Language (XML) [19] 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 [14]. 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 [2]:

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.

2.8 Document Object Model (DOM)

W3C's Document Object Model (DOM) [18] defines an object-oriented API for HTML and XML documents which a Web client can present to programs that need to process the documents [14]. 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.

3 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):

PHASE I: The resource discovery agents working in the background find out about the resources providing products and services. If the resources want to join the marketplace, the marketplace provides them a template workflow of a selling agent. If the resource already has a selling agent, this one is registered to selling trader through Trading Object Service.

We expect resources to expose their semantics by using the Resource Description Framework (RDF) [12] and the Extensible Markup Language (XML) [19]. As briefly summarized in Section 2.6, 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.

The buying and the selling agents which are autonomous, reactive and proactive with negotiation ability, should be defined as workflows since they

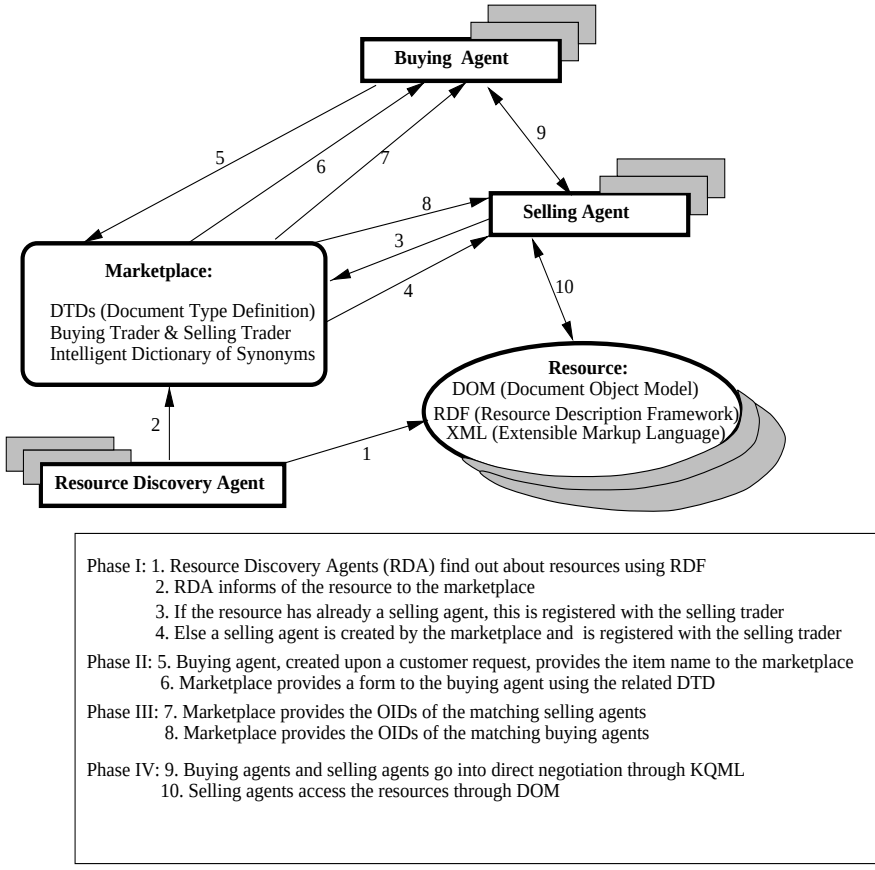

Fig. 1. The Architecture of METU-EMar

consist of processing steps with data and control flow among them communicating with resources, with the customer and among themselves. A workflow system to be used in modelling a buying or a selling agent should have the following properties:

– The scheduler of these workflows must be truly distributed in the sense that the workflow should be able to execute in any node of the network without consulting to a top level central control [5,6]. This is essential since the domain of the workflow contains all the Web resources registered to the marketplace. Also, since the distribution infrastructure is CORBA, these resources must have an ORB. It should be noted that it is possible for different ORBs to communicate through IIOP.
The other components of the workflow, like the history management used for logging and recovery purposes, should also be handled in a distributed way to exploit the advantages brought by a distributed scheduler [9].

- The workflow should be defined as a template that can be enriched or reduced (skipping parts of the prespecified workflow skeleton) so that the customer or the resource can adapt it to its capabilities and requirements. Run time modifications to the workflow should also be possible since improvisations may be necessary in the negotiation phase among the agents.
- The buying agents and the selling agents may include other subprocesses, for example the payment process for the buying agent and the shipment process for the selling agents. In other words independently designed workflows (such as payment workflow) may have to be added to the agents' workflow as subprocesses and therefore the workflow specification method must support this kind of composability [15].

PHASE II: When a customer specifies a service or a product s/he wishes to purchase from the marketplace, a buying agent workflow template is created for the customer. The buying agent is registered to the buying trader through Trading Object Service. The buying agent contacts the marketplace and obtains a form which contains the names and types of the properties of the item. The marketplace uses Document Object Model (DOM) [18] to access the related DTD to obtain names and types of attributes of the product to prepare the form containing this information to be given to the customer.
Document Type Definitions (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 2.7. 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). Marketplace contains references to the DTDs and uses the Document Object Model to access and manipulate parsed DTDs as a collection of objects.
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 marketplace. 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.

PHASE III: The buying agents and the selling agents find out about each other through the related trader objects. Having two trader objects (buying and selling) makes the process symmetric, that is, both buying objects and selling objects can locate all the related agents as soon as they join the marketplace. A buying agent may contact all related selling agents, to determine a 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 goal. The buying agents are on the customers' side and the selling agents are on the resources' side.

PHASE IV: The buying agents go in direct negotiation with selling agents provided by the marketplace. In this respect, RDF is used in encoding resources and query capabilities and KQML [10] 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 [3] 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 [1].

When CORBA and Web is used as the distribution infrastructure, all the agents in the system can be implemented as CORBA objects. Document Object Model defines its interfaces already in IDL which makes it possible to access the resources as CORBA objects, too. Furthermore, using CORBA as the infrastructure provides the opportunity to use OMG's Trading Object Service as a part of the marketplace. 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.

As an extension to this scenario, the buying agent can be activated from an application program through the API of the buying agent. Note that the application might itself be a workflow. In this case, the application program should be designed to be able to fill in the form produced by the marketplace.

3.1 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)[14]. A large US project aims to define specific attribute names for specific elements in computer industry that can possibly be implemented through XML DTDs [4].

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 [1] to better exploit the scheme described.

3.2 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.

4 Related Work

One of the earliest examples of an electronic marketplace is Kasbah [3] 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.

Kasbah has a simple prototype implemented in CLOS using Harequin Lisp to test the basic concepts of negotiation. In Kasbah, all agents are locally built and thus are made to communicate via a predefined set of methods.

A CORBA based electronic broker (OFFER) is described in [1]. 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 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.

5 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 [17].

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. The realization of this architecture depends on the availability of DTDs for different user groups. We also present the modifications to the architecture when DTDs are not available.

References

1. 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.

2. Bosak, J., "XML, Java, and the Future of the Web", http://sunsite.unc.edu/pub/sun-info/standards/xml/why/xmlapps.html.
3. 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.
4. Danish, S., Personal Communication.
5. 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 [6].
6. Dogac, A., Kalinichenko, L., Ozsu, T., Sheth, A., (Edtrs.), "Advances in Workflow Management Systems and Interoperability", Springer-Verlag, 1998.
7. Doorenbos, R. B., Etzioni, O., Weld, D. S., "A Scalable Comparison-Shopping Agent for the World- Wide Web", ACM Agents '97 Conference, 1997.
8. Finin, T., Labrou, Y., Mayfield, J., "KQML as an agent communication language", in Jeffery M. Bradshaw, editor, Software Agents, MIT Press, 1995.
9. Koksal, P., Arpinar, S., Dogac, A., "Workflow History Management", ACM Sigmod Record, Vol. 27, No. 1, March 1998.
10. 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.
11. Lassila, O., "RDF Metadata and Agent Architectures", http://www.objs.com/-workshops/ws9801/papers/paper056.html.
12. 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/.
13. Manola, F., "Towards a Web Object Model", http://www.objs. com/OSA/wom.htm.
14. Manola, F., "Towards a Richer Web Object Model", ACM Sigmod Record, Vol. 27, No. 1, March 1998.
15. 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.
16. Orfali, R., Harkey, D., "The Essential Client/Server Programming with JAVA and CORBA", John Wiley, 1997.
17. Tanenbaum, J. M., "Eco System: An Internet Commerce Architecture", IEEE Computer, Vol. 30, No. 5, May 1997.
18. Document Object Model (DOM), http://www.w3.org/DOM/.
19. Extensible Markup Language (XML), http://www.w3.org/XML/.
20. OMG's Trading Object Service. OMG Document orbos/96-05-06, Version 1.0.0, May 10, 1996.
21. Resource Description Framework (RDF), http://www.w3.org/Metadata/RDF/.
22. Woolridge, M., Jennings, N, "Intelligent Agents- Theory and Practice", Knowledge Engineering Journal, June 1995.