A Multi-Agent System Infrastructure for Software Component Market-Place: An Ontological Perspective

Rıza Cenk Erdur
Dept. of Computer Engineering
Ege University
35100, Bornova, Izmir, Turkey.
Tel:+90 232 3887221 - 216 ext.
erdur@bornova.ege.edu.tr

Oğuz Dikenelli
Dept. of Computer Engineering
Ege University
35100, Bornova, Izmir, Turkey.
Tel:+90 232 3887221 - 236 ext.
oguzd@staff.ege.edu.tr

ABSTRACT
In this paper, we introduce a multi-agent system architecture and an implemented prototype for software component market-place. We emphasize the ontological perspective by discussing the ontology modeling for component market-place, UML extensions for ontology modeling, and the idea of ontology transfer which makes the multi-agent system to adapt itself to the dynamically changing ontologies.

Keywords
Multi-agent systems, component market-place, ontology.

1. INTRODUCTION
As a result of the research and development on "Component Based Software Engineering", software developers have reached on a consensus that component based development is the right way to build extensible, maintainable, high quality, and Web based enterprise applications. Standardization efforts on component models, integration platforms, and business domain concepts based on XML will accelerate the usage and spreading of components for building component based Web architectures. Hence, it can be expected that in a very near future, there will be thousands of open-market components available on the Internet and in such a market-place the problem will be finding the component that fits to the requirements and with the cheapest price.

In the literature, there are some proposals for component retrieval over the Internet, which either uses the search engine approach [10] or the classical data integration techniques [1]. But it is clear that such a massive market-place will necessitate more advanced approaches to create seamless supply chain from component producers (suppliers) to reusers (consumers) in both initial supply and ongoing support.

Autonomous characteristics of all complex behaviour in a component market-place such as initiating a component search depending on reusers' preferences, participating in auctions, and managing component versions over Internet point us the autonomous agents and the multi-agent systems which have proven their ability to model such complex systems with autonomous behaviour. Hence, in this paper, we introduce a FIPA-compliant multi-agent system architecture and an implemented prototype for component market-place, by emphasizing the ontological perspective.

During the development of this system, well-known agent development methodologies such as GAIA [11] and MASE [3] are used to model agent roles and interactions. These methodologies do not propose anything to model explicit ontologies. However, it is impossible to build any open and complex multi-agent system without defining explicit ontologies, which provide the agents a common language for communication. Therefore, in the design phase of the development, we have defined two different explicit ontologies for component market-place, one for modeling component libraries and one for modeling components in specific domains.

Ontologies of the proposed market-place architecture have been defined using UML (Unified Modeling Language), because UML is a well-known modeling language standard and is widely used in software community. Our study is not the only one which uses UML to model ontologies, there are others in the agent community [2]. Although UML's static structures and OCL (Object Constraint Language) are enough for modeling simple ontologies, we have identified

some extension requirements to make UML more proper for modeling complex ontologies such as the component domain ontologies. One of such extensions is for defining rules on associations. Another extension is for defining the hierarchical term spaces of attributes. We have introduced a rule and a term space stereotype for these extensions. These stereotypes and the necessity of these extensions are discussed during the explanation of the component market-place ontologies in section 3.

While developing the multi-agent system architecture for component market-place, we have realized that the agents need to adapt themselves to the dynamically changing and/or newly added domain ontologies. To solve this problem, we introduce the idea of ontology transfer. To transfer ontologies, an XML based content language has been developed. The idea of transferring the ontologies, which is discussed in more detail in section 4, is not specific for component market-place, it can be used in any multi-agent system with dynamically changing explicit ontologies.

2. A MULTI-AGENT SYSTEM ARCHITECTURE FOR SOFTWARE COMPONENT MARKET-PLACE

The multi-agent system architecture proposed for software component market-place is given in figure 1. As it is shown in figure 1, the proposed architecture consists of an agent providing the agent management service, a directory facilitator, an ontology agent, reuser agents, and component agents.

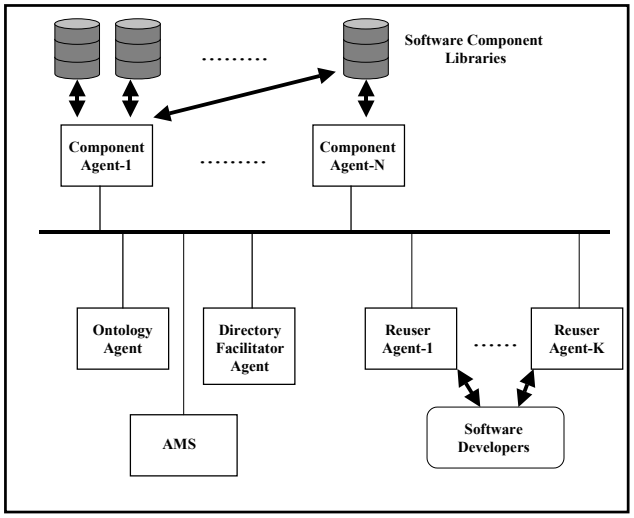

Figure 1. The architecture of the multi-agent system for software component market-place.

The multi-agent system that we have designed for software component market-place is FIPA-compliant [6]; hence, it includes an agent providing agent management service. This agent is called as "AMS" shortly. AMS is responsible for managing the operation of the agent platform and performs the management functions, which are described in the FIPA agent management specification [5]. Registering with the AMS, deregistration from the AMS, and searching/modifying the agent descriptions stored in the AMS are examples for the management functions.

Another necessity for being a FIPA-compliant architecture is the existence of a directory facilitator agent, which is shortly called as "DF". The DF stores the descriptions of the services offered by other agents and provides yellow pages services to other agents [5]. Each agent registers with the DF by advertising the services that it offers. The FIPA agent management specification defines a generic service description template for advertising services, but this template is not sufficient for advertising the capabilities of component libraries. The reason behind this is that the component agents in our multi-agent system advertise their component libraries' capabilities using a specific ontology, rather than using simple tuples consisting of some keywords. We call this specific ontology as the "Component Library Advertisement Ontology". Using a specific ontology for advertising services to the DF does not conflict with FIPA-compliance, because FIPA agent management specification allows information other than the standard service descriptions, to be included in the DF.

The component library advertisement ontology is not the only ontology used in the software component market-place. There are also domain ontologies, with which the component meta-knowledge of the component libraries in different domains is modeled. Component meta-knowledge is the extra knowledge about a component in a specific domain. It makes the components easy to understand and reuse. Since there are many component libraries providing components in different domains in the market-place, there are also many domain ontologies for these different domains.

The ontology service is provided by the ontology agent. The ontology agent is autonomous in the sense that it initiates a task for sending each updated ontology to the related agents. The ontology agent can also send a specific ontology to other agents whenever requested using FIPA-request protocol. The domain ontologies, the component library advertisement ontology, and the standard FIPA agent management ontology are stored in the ontology agent. The ontology agent controls all changes on the ontologies. The ontology agent informs all the related agents when an ontology is updated or a new domain ontology is added. The related agents then transfer the updated or newly added ontology using an XML based content language, which can transfer ontologies [4]. Ontology transfer is discussed in more detail in section 4.

The reuser and component agents are specific for the software component market-place application. We have defined their responsibilities in the analysis phase of the multi-agent system development methodology that we have

used [3][11]. We will discuss their basic responsibilities shortly below.

The reuser agents are the reuse based software developers' interface to the multi-agent system. They have the basic responsibilities such as assisting software reusers in formulating queries about component libraries and component meta-knowledge, presenting query results to the reusers in their specific presentation format, initiating a component search and retrieval process automatically after learning reusers' preferences, and participating in software component auctions as the component buyers.

Component libraries are wrapped with the component agents so that they can be integrated into the agent based component market-place environment. The component agents hide the internal access mechanisms of the component libraries from the rest of the system. They answer one-shot or periodic component meta-knowledge queries coming in specific domain ontologies. They do the version management. For example, when a new version of a component is produced, all reuser agents, who had previously retrieved the old version of that component, are informed by the component agents. Participating in software component auctions as the component sellers is another basic responsibility of the component agents.

2.1. Interactions for Component Retrieval in the Market-place

The component retrieval in the proposed multi-agent system is a two-step process. In the first step, the component library advertisements, which are stored in the directory facilitator agent (DF), are searched using the component library advertisement ontology, to discover the related component agents. The DF, which is in fact a matchmaker, returns the names and addresses of the related component agents to the reuser agent. The reuser agent stores this knowledge as part of its coordination knowledge. This step, which is a FIPA-request interaction, is shown in figure 2 as an AUML diagram [9].

The component library advertisement ontology is used in the first step of component retrieval process. Using this ontology, reusers formulate a query to be sent to the DF, to discover the component agents, which are wrapping the component libraries providing components in a specified domain and type. Then, the addresses of the related component agents are returned to the reuser agent by the DF.

In the second step, the component agents, which are discovered in the first step, are queried for the component meta-knowledge descriptions. These queries are formulated using the specified domain ontology's query interface. If a specific domain ontology is not stored in the reuser agent's knowledge base, it sends a request to the ontology agent for transferring that ontology before the query is formulated. It maps the transferred ontology into its local knowledge base and creates the query interface related with that ontology. The query is formulated using this interface. A number of component meta-knowledge descriptions are returned as the result of this query.

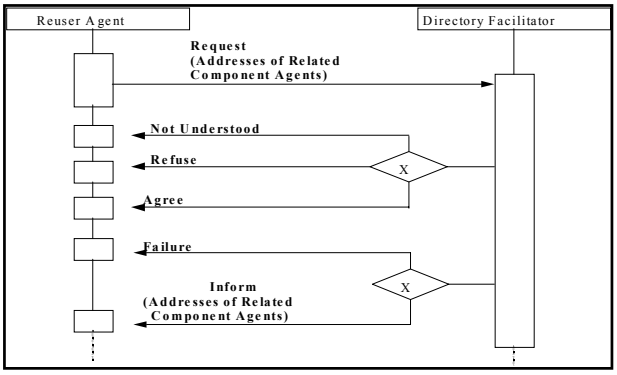

Figure 2. First step of the component retrieval process.

The human reuser inspects the component meta-knowledge descriptions, decides for which components he or she will initiate an auction, and informs the reuser agent. Reuser agent extracts the minimum price from the meta-knowledge descriptions of all selected components and sets it as the price quote. Then, it initiates an auction using the FIPA Dutch Auction Interaction Protocol. Related component agents join the auction as participants and they propose new bids lower than the current price quote until the lowest possible price, which is specified during the initialization of each component agent, is reached. At the end of the auction, reuser agent buys the component with the lowest bid price.

Assuming that the specified domain's ontology has already been stored in the agent's local knowledge base, the interaction for component meta-knowledge retrieval is shown in figure 3 as an AUML diagram.

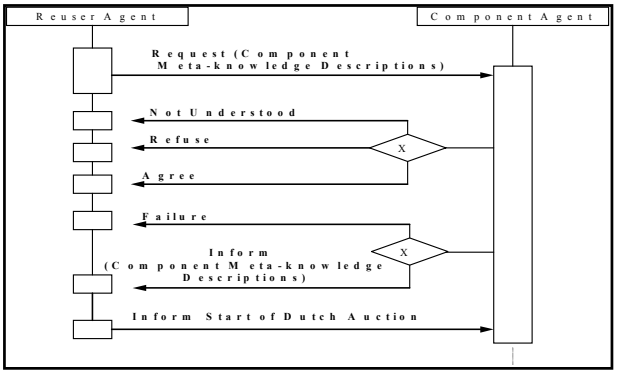

Figure 3. Second step of the component retrieval process.

During these interactions different ontologies are used. These ontologies are explained in detail in section 3.

3. ONTOLOGIES FOR SOFTWARE COMPONENT MARKET-PLACE

Different ontologies are used during different agent interactions in an agent based component market-place environment. The standard FIPA agent management ontology is used in interactions between the agent management service and the other agents. The component library advertisement ontology is used in interactions between the directory facilitator agent (DF) and the other agents. The reuser agents and the component agents interact using different domain ontologies. The component library advertisement ontology and the domain ontologies used for modeling component meta-knowledge in different domains are explained in detail in the following subsections. The FIPA agent management ontology will not be explained, since it is a domain independent standard ontology, which is described in detail in the FIPA agent management specification [5].

3.1. The Component Library Advertisement Ontology

The component library advertisement ontology, with which the component agents advertise their component libraries' capabilities to the DF, is important in the sense that it also classifies the component libraries. Such a classification is needed, because there may be thousands of component libraries providing different types of components in different domains in the market-place.

Part of the component library advertisement ontology is shown in figure 4 in UML notation.

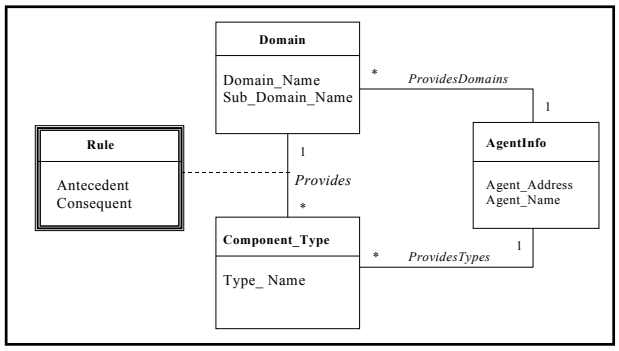

Figure 4. Part of the component library advertisement ontology.

UML has constructs such as classes, attributes, and relations that can be used for modeling ontologies. It also includes some extensions such as the Object Constraint Language (OCL) and stereotypes that assist in representing ontologies. OCL can be used to define constraints on different elements of the model. Stereotypes are extension mechanisms that are used to define new modeling elements that can have distinct semantics, characteristics, and notation.

In our case, stereotypes have been used for adding rules to the ontology and for defining term spaces on an attribute of an ontology. Rule stereotype, which is extended from the UML static element “Class”, marks the associations. A rule stereotype, which marks the “Provides” association, is shown in figure 4. This stereotype consists of an antecedent and a consequent. The antecedent may include one or more clauses, while the consequent includes an operation to be activated. Clauses are separated by "And" or "Or" operators. Each clause consists of a rule variable, an operator, and a value. For example, a rule which says that "when design pattern components in multimedia domain are queried, the framework components should also be returned as a result", can be expressed in our specific rule representation format as "IF Domain.Domain_Name=Multimedia AND Component_Type.Type_Name=Design_Pattern THEN addQuery(Component_Type.Type_Name=Framework)". In this example, the antecedent contains two clauses separated by "AND". The "Domain.Domain_Name" is the rule variable in the first clause, while the "=" is the operator, and "Multimedia" is the value. The rules are interpreted by a specific interpreter during the query execution, and the queries are expanded or modified using the rules.

We have not used OCL in representing the rules because OCL expressions cannot activate operations. On the other hand, OCL can be used to define first order predicate constraints on the elements of the ontology. The truthness of these constraints can be checked by using an OCL interpreter at runtime.

The term space stereotype, which is extended from the UML static element “Attribute”, models the controlled vocabularies represented hierarchically in a graph structure, which holds the similarities between terms. This stereotype is an important element for an ontology, since controlled vocabularies have an important role in ontology modeling. We have used the term space stereotype to model the "Domain_Name", "Sub_Domain_Name", and "Type_Name" attributes of the classes "Domain" and "Component_Type".

The association relations between each of the classes in the component library advertisement ontology are shown in figure 4. The association named as "Provides", specifies which domains provide which types of components. The other associations named "ProvidesDomains" and "ProvidesTypes" specify the domains and the types of the components provided by the component agents.

The attributes of the classes "Domain" and "Type" take values from a controlled vocabulary. For example, the "Domain_Name" attribute of the class "Domain" may take values from a set V_1={Multimedia, Telecommunication, Education, Finance, ...}, while the "Type_Name" attribute of the class "Component_Type" may take values from a set V_2={Analysis_Pattern, Design_Pattern, Source_Code,

Framework, EJB, ...}. These controlled vocabularies are modeled using the term space stereotype as mentioned above.

3.2. Domain Ontologies

To be able to understand and reuse the components easily, some extra knowledge about the components is needed. This extra knowledge is called as the "component meta-knowledge". The component meta-knowledge for different domains is modeled by the domain ontologies. The domain ontologies play an important role in component market-places, because they make the software developers and the component providers to talk to each other in the same language.

In software reuse literature, some templates have been proposed for component meta-knowledge modeling. The BIDM (Basic Interoperability Data Model) by RIG (Reuse Library Interoperation Group) [8] is one of the well-known data models that is defined for modeling component meta-knowledge. The BIDM defines a generic template for meta-knowledge description. This template is extended using our rule and term space stereotypes so that it can be used as a generic template for defining different domain ontologies in the market-place.

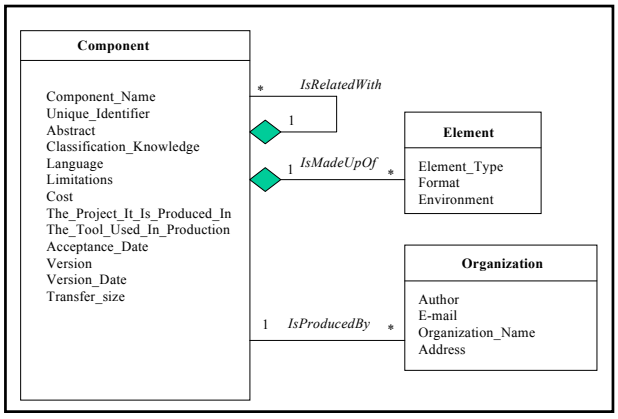

Figure 5. Part of the domain ontology definition template.

Figure 5 shows a part of the domain ontology definition template. The domain ontologies for different domains are defined by specializing this ontology definition template. Some of the attributes of the class “Component” are common in all ontologies and some may change. For example, the "Classification_Knowledge" attribute, which is used for classifying the components in a specific domain using a specific scheme, should be defined separately for each different domain. Also, the "Component_Name" attribute may change in different domains, since each domain may have its own naming scheme. However, the attribute specifying the language of the component has a controlled vocabulary, which is common in all domains and constructed using our term space stereotype. The "Element" class specifies the sub elements of a component. For example, a source code component may have a user manual document and a technical document sub elements.

4. ONTOLOGY TRANSFER

An important problem in multi-agent system based component market-places is to make the agents to adapt themselves to the dynamically changing domain ontologies. For example, any change in telecom domain ontology must be known by all component agents that supply telecom components to the market-place. Our approach for solving this problem is to give the agents the ability of transferring the ontologies at run time. By this way, when the ontology agent informs an update and/or a newly added ontology, the related agents request for transferring this ontology from the ontology agent. The ontologies are transferred using an XML based content language [4]. The developed content language is not only used for transferring ontologies, but also for advertising services, sending queries, and returning of results in a multi-agent system. The document type definition (dtd) for this XML-based content language is not given because of space limitations, but to give an idea about how it transfers classes, attributes, and relations of an ontology, some parts of an example content is shown in figure 6.

```
<Content>
 <Action>
   <Class cnameID="agent_identifier">
    <Slot sname="name" svalue_type="String">
    <SlotValues>OA@reuseproject</SlotValues></Slot>
    <Slot sname="address" svalue_type="String">
    <SlotValues>//155.2.3.13:1234/OA@reuseproject
    </SlotValues> </Slot> </Class>
   <Function func_name="ontology_transfer">
    <Ontological_Description>
    !-- Tags to transfer classes, attributes, relations and rules
    <Ontology_Name>MultimediaDomain</Ontology_Name>
    <Class cnameID="Component">
     <Slot sname="Component_Name" svalue_type="String">
     <SlotValues>Show_Video_Clip, Add_Video_Clip,
         Resize_Video_Clip, Synchronize_Sound_&_Video
     </SlotValues></Slot>
            :
    </Class>
       :
     <Relation>
       <Association>
           <End1>IsProducedBy</End1>
           <Multiplicity>1,*</Multiplicity>
           <Element1>Component</Element1>
           <Element2>Organization</Element2>
       </Association>
     </Relation>
        :
    </Ontological_Description>
   </Function>
 </Action> </Content>
```

Figure 6. Ontology transfer in XML content.

This example shows how to transfer an ontology in multimedia domain. Classes, attributes, term spaces, relations, and rules of an ontology are transferred within their own tags inside the "Ontological_Description" tag. For example, the controlled vocabulary belonging to the "Component_Name" attribute of the class "Component" is transferred within the "*<Slot sname="Component_Name" svalue_type="String"> <SlotValues>Show_Video_Clip, Add_Video_Clip,Resize_Video_Clip,Synchronize_Sound_& _Video </SlotValues></Slot>*" tag. In the current implementation, the ontology dependent queries and results are also transferred inside the "Ontological_Description" tag. In the next prototype, we plan to use the XQuery standard for ontology dependent queries. Since the developed content language is used in FIPA agent platforms, it contains other tags such as action and function to support the standard FIPA agent management ontology, which is used in communicating with the AMS and DF. For example, the "agent_identifier" class under the "Action" tag specifies the agent from which an action is requested. The "Function" class specifies the type of the action. In this case, the requested action is ontology transfer. The ontologies are mapped to the agents' local knowledge bases after being transferred. This approach brings in the following benefits:

1. The ontology dependent query interfaces are created dynamically using the ontologies stored in agents' local knowledge bases.
2. Ontologies can be personalized if they are stored in local knowledge bases. Each agent can extend the stored ontologies by adding extra local rules. The personalized ontologies can then be used locally in expanding or modifying queries that an agent has to execute.
3. By transferring ontologies, ontological knowledge becomes distributed to the overall system, which makes the agent system more robust and reliable than an agent system which has only one central ontology agent responding to all ontological requests such as the one described in the FIPA ontology service specification [7].

5. CONCLUSION

To realize the proposed architecture and to evaluate our ideas about ontology transfer, we have constructed a multi-agent system prototype for component market-place. During the construction of this multi-agent system prototype, we have used the agent framework, which we had previously developed [4]. Different than others, this framework has an extra layer, which contains the reusable behaviour related with ontology transfer; hence, the agents constructed by using it, inherit the ability for ontology transfer. It also has a built-in support for the basic auction protocols, such as English or Dutch auctions. To evaluate our ideas about ontology transfer, we have added new domain ontologies and/or changed some of the existing domain ontologies at run time. We have shown that the agents in the prototype multi-agent system can adapt themselves to the dynamically changing and/or newly added ontologies, by transferring the related ontologies and by mapping them to their local knowledge base. Instantiating the agents using a framework like the one we had already developed, which has a built-in support for ontology transfer related actions, seems to be an important requirement for the multi-agent system based component market-places.

6. REFERENCES

[1] Braga R.M.M., Mattoso, M., and Werner, C.M.L. The use of mediation and ontology technologies for software component information retrieval. Symposium on Software Reuse (SSR'01), Toronto, Canada, 2001.

[2] Cranefield, S., Haustein, S., and Purvis, M., UML-based ontology modeling for software agents, in the Proceedings of the Workshop on Ontologies in Agent Systems held at the 5th International Conference on Autonomous Agents, 2001, Montreal, Canada. Available online at http://cis.otago.ac.nz/OASWorkshop.

[3] Deloach, S.A., Wood, M.F., and Sparkman, C.H. Multiagent system engineering. International Journal of Software Engineering and Knowledge Engineering. Vol.11 No.3 (2001) 231-258.

[4] Erdur, R.C. and Dikenelli, O., The design and implementation of a FIPA-compliant framework with a new layer for ontological behaviour, EMAG-2001-02, Technical Report, Ege University, Dept. of Computer Engineering, 2001, Bornova, Izmir, Turkey.

[5] FIPA(a). FIPA XC00023H: FIPA agent management specification. http://www.fipa.org, 2001.

[6] FIPA(b). FIPA XC0001J: FIPA abstract architecture specification. http://www.fipa.org, 2001.

[7] FIPA(c). FIPA XC00086D: FIPA ontology service specification. http://www.fipa.org, 2001.

[8] IEEE. IEEE standard for information technology - data model for reuse library interoperability: Basic interoperability data model (BIDM), 1996, USA, IEEE Std:1420.1-1995, ISBN:1-55937-584-1.

[9] Odell, J., Parunak, H.V.D., and Bauer, B. Extending UML for agents. in the Proceedings of Agent Oriented Information Systems Workshop at AAAI '2000, Wagner, G., Lesperance, Y., and Yu, E. (eds.), Austin, TX, 3-17.

[10] Seacord, R., Hissan, S., and Wallnau, K. Agora: A search engine for software components. Technical Report. CMU/SEI-98-TR-011, August, 1998.

[11] Wooldridge, M., Jennings, N., and Kinny, D. The Gaia methodology for agent-oriented analysis and design. Journal of Autonomous Agents and Multi-agent Systems. Vol. 3(3), 2000.