Virtual Laboratory – Agent-based Resource Sharing System

Goran Kimovski Vladimir Trajkovic Danco Davcev
Faculty of Electrical Engineering and Computer Science,
Ss. Cyril and Methodius University, Skopje, Macedonia
goran.kimovski@seavus.com; {trvlado; etfdav}@cerera.etf.ukim.edu.mk

Abstract

In this study, we extend the concept of distance education and our Virtual Classroom (VC) with adding a new service, that we call Virtual Laboratory (VL). While the VC represents an interface between the students and a virtual professor and provides personalized learning materials to the users, the VL offers a possibility to the attendees to share different resources out of time and space boundaries. It enables geographically separated users to effectively facilitate remote access to various, presumably diverse, (real) resources.

The VL is implemented with three key points in mind: modularity, reusability and common interface handling. Using object-oriented approach in the development of the VL service, we made a modular system, capable of managing various resources that can possibly be shared through VL. By using common interfaces that each resource-specific agent must implement, many different resources that possibly have nothing in common can be shared through the same VL service.

The first experiments show that users are satisfied with the VL usability. They found VL as a very convenient service within the Distance Education Systems.

***Keywords**: object-oriented design, distance education systems, agents, Java, DCOM, XML*
***Technical area**: Distributed and intelligent objects and agents*

1. Introduction

The scope of Internet services continues to expand, stretching to new fields and encompassing increasingly human activities in the virtual world of the web. The possibility of contacting people and sharing information with the rest of the world extends the usability of this media to a wide variety of users, from the computer experts to the ordinary people looking for entertainment. New concept of education on distance was introduced, using Internet as a mediator between students and trainers as well as a valuable source of information, literature and other resources. The concept becomes more attractive due to the possibility of taking classes, searching virtual libraries, contacting trainers and other colleagues, using only a PC connected to the net, without the necessity of physical attendance at one place and time as is in the classical concept of education. Virtual institutions (e.g. Universities) can be defined as representations of electronic workplaces that enable better exchange of personalized learning material, administration material and provide unique debating fields for students that are showing interest for it.

Virtual organization model can be used to serve as general model for implementing framework for this kind of institutions [1]. The general idea of this model is that the management of goal-oriented activity is independent of the means for its realization and can be realized by switching among optional requirement satisfiers. A way for providing

0-7695-1251-8/01 $10.00 © 2001 IEEE

successful management that reduces the cost of switching among optional educational materials is the use of software agents.

In the context of the work presented n this paper, we see agents as a community of collaborative, autonomous software entities cooperating among themselves in order to accomplish a task, rather than separate entities attempting to expose some anthropomorphic behavior. Cooperation among agents is established through the act of exchanging messages. Different types of agents have different specialization, thus encapsulating different services, but they still have similar features. Agents act independently, performing their work with no importance on how it is done as long as it gives satisfactory results within the software architecture context.

In this study, we extend the concept of distance education (the Virtual University found in [2]) and our Virtual Classroom [3] (VC) with adding a new service, that we call Virtual Laboratory (VL).

The VC service represents the interface between the students and a virtual professor and provides personalized learning materials to the users. Personalized learning materials are created by system adaptation to the student needs shown in the student's interaction process. That process is supported with the help of several mobile agents with different specialty, such as an agent responsible for student classification and a set of agents responsible for supporting learning activities such as finding the learning material relevant to student's needs.

The VL system represents a collaborative software environment that enables geographically separated users to effectively facilitate remote access to various, presumably diverse, (real) resources. It represents an extension, a new service, to the VC concept. Namely, the concept of VL is to offer a possibility to the attendees to share different resources outside the time and space boundaries (breaking the time boundary doesn't always mean concurrent access of different users to the same resource at the same time; it highly depends on the resource nature). It tends to provide remote users an experience that approaches the same as "being there". An example of a possible scenario can be to control remotely a robot system in a chemical laboratory from a PC connected on Internet.

The general idea of the system is to enable sharing of many different resources between as many users as possible. The system is structured in common software framework building blocks or "middleware" to enable the user-resource collaboration to succeed. The idea of a resource is not limited to some laboratory equipment. It can be some software application or some interface to data stored in a remote database for example. In this way, we are not concerned with the nature of the resources that this system should work with. On the other hand, it adds another level of complexity, since we have to take into account that the system can be used for resources that possibly have nothing in common.

In order to satisfy these requirements, when designing the system, we used the object-oriented approach. While designing the system we had three key points in mind: modularity, reusability and common interface handling. The last point was the most critical, since it was the main goal of the system, to allow many kinds of resources that possibly have nothing in common to be shared via the same VL service. Therefore, we defined common interfaces, which are supposed to be implemented by each object (i.e. agent) connected to a specific resource, enabling the system to be concerned with the user-resource interaction and not the nature or implementation details of the resource shared through it.

As with VC, we use agents in the VL design as well. The difference is that they are stationary agents and do not expose any mobility behavior. Different types of agents have different specialization, thus encapsulating different services, but they still have similar features. They act independently, cooperating amongst themselves through the act of exchanging messages. In the case of the VL service, we designed several protocols for socket

communications between the agents of different levels and different hosts in the system. Having in mind the features and increased implementations in XML (Extensible Markup Language), we decided to define all communication protocols within VL using XML. It proved our choice, providing a great deal of flexibility, which is very important for the service, since it brings many different resources into one standard interface.

The following section 2 presents the related work. Section 3 gives the design of the VL service. The discussion about the implementation is given in section 4 and the first experimental results are presented in section 5. Section 6 concludes the paper.

2. Related work

The need for domain analysis, interchange ability of components, and common architecture definition is addressed in [4].

Self-learning agents used for collection and selections of Web pages are presented in [5]. Multiple agents coordinate activities without explicit communication. They use information from user profiles in order to recommend items unseen by other users. Agents learn from their users and they are collecting Web pages rated as better then pages from competing sources.

Designing multi-agent systems that bring assistance into normal work environment, while relieving users of lower level administrative and clerical tasks, is given in [6]. Agents are used as supervisors of students. They process information related to them, distribute it among them and obtain feedback from them.

The role of the software agents that support distance education systems within virtual institutions and their specialized application in that context are mentioned in [7], [8]. The primary purpose of the software agents in these environments is for retrieving information.

An overview of resource discovery systems and their approaches is given in [9]. In [10], a model for type- specific, user- customizable information extraction and a system implementation called Essence is presented. It generates file summaries that can be used to improve both browsing and indexing in resource discovery systems. In [11], distributed system architecture called Harvest that supports Internet resource discovery systems is presented. For motivation of the research problems addressed by Harvest (of which Essence is one component) see [11] as well. The main objectives of these systems are about resource discovery.

In [12], agent technology is used for developing a Virtual Laboratory that is also used as a tool for researchers at three collaborating universities. A dynamic repository of information on ATM topics, ATM simulators and ATM hardware is among resources being made available.

In our approach, VL has to provide an efficient sharing of resources among distance learning students according to the student profiles. We defined several protocols in XML suitable for communication between agents. In order to provide better functionality of the VL, common software framework building blocks or "middleware" ties all services together. This approach should enable an easy cooperation among distributed teams as well.

3. Virtual classroom and laboratory design

VC is defined as an alternative of the "classical classroom". It can be used as an additional source for materials presented in certain courses. The main idea in the VC design is that users can log on from anywhere, and that on the user's end, the only support necessary, is a Web

browser. The whole system is web based and uses Java applets for its implementation. The logical model of the VC system is given in [3].

The VC system lacks a mechanism for offering a remote laboratory exercises possibilities. In order to overcome this deficiency, a new system called VL has been developed and integrated with the VC system. It should offer a complete distance education system that offers possibilities to the students to follow courses, make tests and exams, communicate with real professors, remotely work with a laboratory instrumentation and equipment etc.

The VL service may be considered within the virtual laboratory defined as an alternative to the "classical laboratory", but it is obvious that this service is quite general. It offers universal, ubiquitous, easy access to any type of remote resources in a distributed system.

We use the object-oriented design for the VL system in order to accomplish three key points of the VL concept:

- Modularity - different modules should have a dedicated set of agents with a defined set of tasks
- Reusability - possibility to reuse object definitions in the system through several ways: inheritance, aggregation etc.
- Common interface handling - definitions of common interfaces that can handle virtually any type of resource that could be shared over the VL

Figure 1 represents the collaboration diagram of the scenario that runs when the system is servicing a resource request.

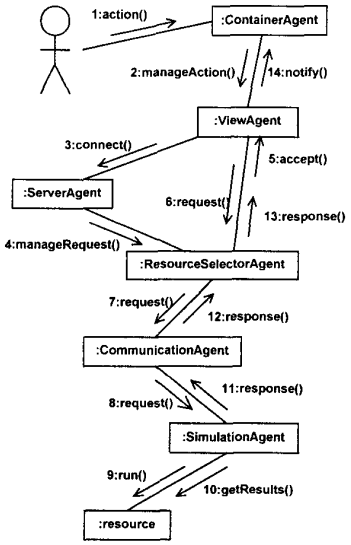

Figure 1. Collaboration diagram

Every user can work with the system by using a web-based interface. The Container Agent enables different resources, with different implementations of their graphical user interface (GUI), to be placed on the web-based interface. In this way every specific resource can be presented with a GUI representing the nature of the resource and can communicate with the users in a manner that is close to the "classical", e.g. having a software compiler as a resource, a source code editor and an output field could represent its GUI. The Container

Agent's task is to initialize the web-based interface by gathering information about the resources available in the system and providing an effective way for the user to select a desired resource.

The Container Agent communicates user's actions to the View Agent, which tries to connect to the Server Agent in order to send the resource request. The View Agents are key players on the client side (the web browser that the user is using to view the VL system). They are responsible for the GUI of the particular resources. The system can hold many definitions of different View Agents, since every different resource or a group of resources should present a GUI of its own to the user.

All of the system communication runs through two agents that run on the VL server machine, the Server Agent and Resource Selector Agent. Every View Agent in the system communicates with the Server Agent when a connection to the VL server is required in order to communicate with a particular resource. The Server Agent handles different connection requests and runs a Resource Selector Agent as a thread that handles the specific requests and passes back the resource responses to the View Agent.

After the acceptance of the connection request by the Resource Selector Agent, all of the requests from the View Agent are passed through the Communication Agent and Simulation Agent to the (real) resource. The Communication Agent's task is to enable the message translation and communication in general with the Simulation Agent and the (real) resource.

After the resource processes the request and acts accordingly, the result is passed back through the same channel to the View Agent that shows the result to the user, by redrawing it self for example.

All of the agents mentioned above are stationary agents and implemented as pure Java classes, but the implementation of the set of Simulation Agents is based on DCOM (Distributed Component Object Model). This enables a distributed computing of the resource requests. The Simulation Agent's task is to wrap around a physical resource in a software entity working in the VL system. The Simulation Agent doesn't always need to be connected to a specific real resource. It can simply be a software simulation of a particular resource.

The whole system is separated in three different modules: view, server and simulation module. Every module contains its own set of agents and other objects. It communicates with the objects in the other modules via XML defined protocols (see an example on Figure 2) that give a possibility to exchange several common messages.

View Agent	Resource Selector Agent
`<CONNECT/>`	
	`<ACKNOWLEDGEMENT/>`

`<ACKNOWLEDGE>`
`<ID>rmClient35</ID>`
`</ACKNOWLEDGE>`	

	`<ACCEPTED>`
	`<ID>rmClient35</ID>`
	`</ACCEPTED>`

`<RESOURCE>`
`<ID>res_Chem_Robot</ID>`
`</RESOURCE>`	

	`<RESOURCE>`
	`<ID>res_Chem_Robot</ID>`
	`<AVAILABLE>YES</AVAILABLE>`
	`</RESOURCE>`

`<REQUEST>` `<COUNT>1</COUNT>`	
	`<CONTINUE>YES</CONTINUE>`

`<ITEM>`
`<TYPE>STRING</TYPE>`
`<NAME>Direction</NAME>`
`<VALUE>LEFT</VALUE>`
`</ITEM>`	

	`<CONTINUE>YES</CONTINUE>`
`</REQUEST>`	
	`<CONTINUE>YES</CONTINUE>`
`<RESPONSE>` `<COUNT>2</COUNT>`	
	`<CONTINUE>YES</CONTINUE>`

```
   <ITEM>
      <TYPE>NUMBER</TYPE>
      <NAME>x</NAME>
   </ITEM>

   <ITEM>
      <TYPE>NUMBER</TYPE>
      <NAME>y</NAME>
   </ITEM>
</RESPONSE>

<CONTINUE>YES</CONTINUE>

<CONTINUE>YES</CONTINUE>

<CONTINUE>YES</CONTINUE>

<DISCONNECT>
   <ID> rmClient35</ID>
</DISCONNECT>
```

```
<CONTINUE>YES</CONTINUE>

<CONTINUE>YES</CONTINUE>

<RESPONSE>
   <COUNT>2</COUNT>

   <ITEM>
      <TYPE>NUMBER</TYPE>
      <NAME>x</NAME>
      <VALUE>135</VALUE>
   </ITEM>

   <ITEM>
      <TYPE>NUMBER</TYPE>
      <NAME>y</NAME>
      <VALUE>23</VALUE>
   </ITEM>
</RESPONSE>

<DISCONNECTED>
   <ID> rmClient35</ID>
</DISCONNECTED>
```

Figure 2. XML defined protocol

Figure 3 represents the class diagram of the agent classes in the system.

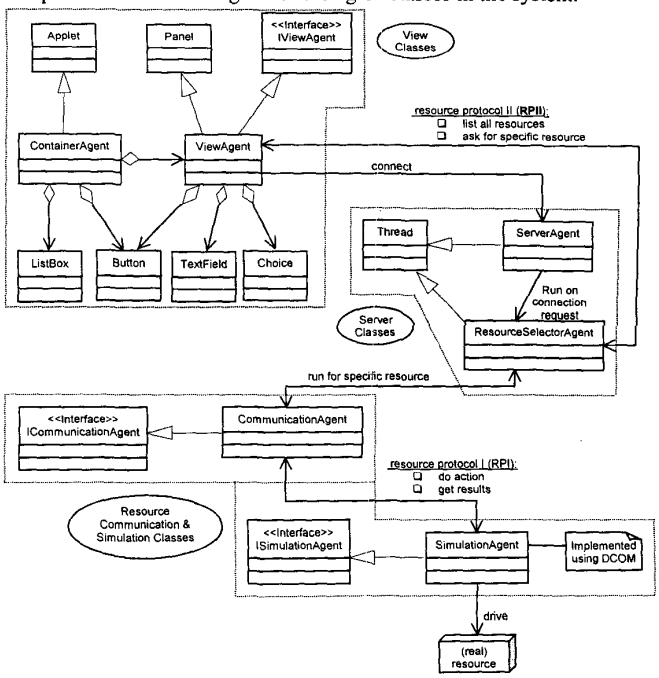

Figure 3. Class diagram of the agent classes

As it can be seen from the class diagram, the Container Agent and View Agent are concerned with the graphical representation and generalize the Applet and Panel classes, which give the ability the user interface of the VL system to be sited in a web page and available via browser on the user side. It is necessary that different View Agents implement a common IViewAgent interface. This interface defines all of the methods and properties that a View Agent object has to implement. The system isn't concerned with the details of the specific implementation for a particular resource.

Server Agent and Resource Selector Agent inherit their behavior from the Thread class and are associated with the View Agent on one side and Communication Agent on the other side.

The same approach as the View Agent with implementing a common interface is used for the Communication Agent (ICommunicationAgent interface) and the Simulation Agent (ISimulationAgent interface). These two agents are working more tightly with the (real) resource than the other agents. They communicate with the Resource Selector Agent in order to service the different requests from the View Agent.

4. Implementation

VL is implemented in Windows 9x/NT environment. All agents are developed in Java, with exception of Resource Simulation Agents, which are based on DCOM technology. Being of stationary type, no mobility support platform is required for the VL service agents.

Using object-oriented approach in the development of the VL service, we made a modular system, which offers easy and flexible way of managing various resources that can possibly be shared through VL. By using common interfaces that each resource-specific agent must implement in order the system to be able to work with them, many different resources that possibly have nothing in common can be shared through the same VL service. Another big advantage is the possibility of reusing already developed agents in order to introduce new agents that can handle resources with additional functions than the basic ones and at the same time not to worry about the implementation of the communication with the system or the implementation of the basic functions the resource supports.

Figure 4 shows the component layout of the VL service.

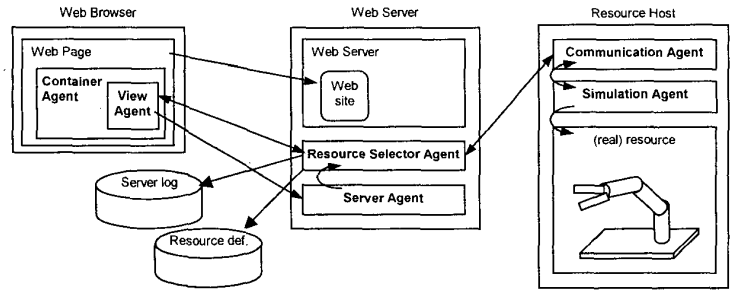

Figure 4. The main components of the VL service

Our system runs on three hosts. The first host is the client machine of the user browsing the VL via Browser. The Container Agent and the set of View Agents reside here, presenting the different resources' GUIs to the users.

The second host is the VL server. It runs the web server software, and Server and Resource Selector Agents. These agents have access to two databases, Server Log database and Resource Definitions database. The Server Log database is used as a log for every event on

the VL server and for the requests history received by the Server Agent and Resource Selector Agent. The Resource Definitions database purpose is to contain information on the resources available in the system. The list of resources available in the system, required by the Container Agent at its initialization is drawn from this database, as well as the information for the correct Communication Agent to which the Resource Selector Agent has to pass the request for the particular resource.

The third host is a computer connected to the (real) resource. The specific Communication Agent and Simulation Agent for a particular resource reside there. As we mentioned before, the Simulation Agent doesn't necessarily have to work with some real resource, it can simulate the resource behavior to the rest of the system.

The next two figures show the implemented VL in action. Figure 5 shows the resource called "Robot Car" under the control of our VL. This resource implements an interface that allows users to control robot movements in six possible directions. The robot car that is attached to a PC in some laboratory can be controlled remotely via the VL web interface by simply clicking the buttons representing different directions. The robot trajectory is drawn on a panel, so the user can track the robot movements. Additionally, a web camera is following the robot movements, so it can be watched on-line while moving. This type of resource can be very handy for allowing the students to have their own walkthrough inside some distant and expensive laboratory, or a laboratory where human access is prohibited.

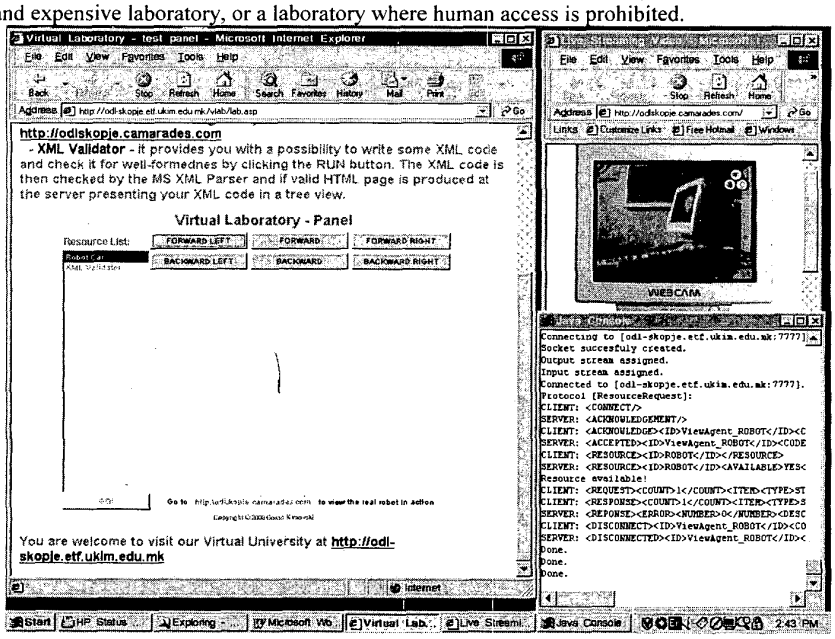

Figure 5. VL in action (resource: Robot Car)

Figure 6 shows a very different resource running in the same framework defined by the VL. This resource represents a software code syntax checker (in this case XML code). The user can type his/her code into the input provided by the resource's GUI and than it will be checked by the Microsoft XML Parser running on a totally different machine. This resource is

more a test resource and has been developed for the purpose of presenting the VL ability to share resources that have nothing in common.

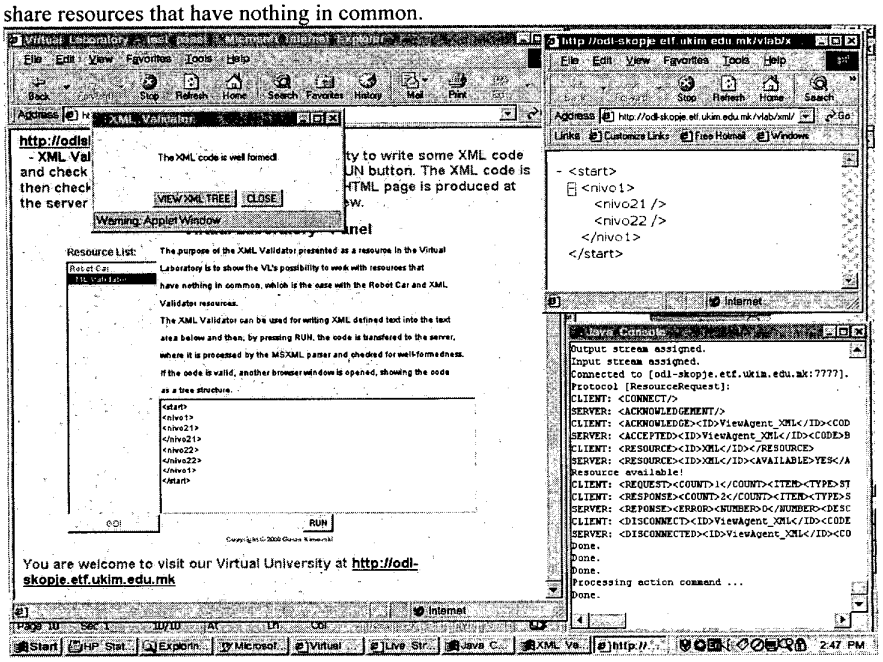

Figure 6. VL in action (resource: XML Validator)

5. Experimental Results

The latest phase of the work on the VL system consisted of performing tests for its functionality, usability and education efficiency.

Functionality tests were performed during the design and implementation of the modules of the system. We involved end users (students) in these phases and used feedback from them to add all requested functionality.

Usability and education efficiency of the system was tested in order to show when and how VL should be used, using participants interview technique.

We involved 23 users from the "Ss Cyril and Methodius" University – Skopje, Macedonia (students from Computer Science, Telecommunications and Electronics), 10 users from the Technical University "St. Kliment Ohridski" – Bitola, Macedonia, and five users from "Politecnico di Torino" – Turin, Italy (38 in total) in the analysis.

In general, users are satisfied with the VL usability, although some of them would like to see more extensive list of resources before they state their conclusion. They found VL as a very convenient service within the Distance Education Systems and suggested various possibilities: immersion, distance control of unsafe laboratory procedures, following the professor that is not in a fixed position during the lectures, inexpensive solution for sharing expensive laboratory equipment, etc.

Since the user interface is different for different resources, some of the users suggested that it could be different for different user groups as well. We found that it is a very good idea, and we will implement it in the next version of our system.

This kind of feedback will continue to contribute to our system. More extensive research and evaluation of the impact of the system on the students' learning process will be performed during the following work on the system. Future work should also address better integration with the VC system in order to connect the courses with the laboratory exercises, but development of various resources and research of their usability for different distance education students should be performed first.

6. Conclusion

In this study, we extended the concept of distance education with adding a new service, that we call Virtual Laboratory (VL).

The designed VL offers a possibility to the attendees to share different resources at once and work with them as if they were at the same place where (real) resources are.

Using object-oriented approach in the development of the VL service, we made a modular system, which offers easy and flexible way of managing various resources that can possibly be shared through VL. By using common interfaces that each resource-specific agent must implement in order the system to be able to work with them, many different resources that possibly have nothing in common can be shared through the same VL service.

Research is needed to identify the most useful resources for distance education students and how the VL service should be best integrated with the Virtual Classroom service.

More extensive research and evaluation of the impact of the system on the students' learning process will be performed during the following work on the system.

7. REFERENCES

[1] Mowshowits, "Virtual Organization", Communications of the ACM, Vol. 40, No 9,1997, pp.30-37;

[2] D.Davcev, V.Cabukovski, "Agent-Based University Intranet and Information System as a Basis for Distance Education & Open Learning", In. Proc. of the 1st UICEE Annual Conf. on Engineering Education, Monash University, Clayton, Melbourne, February 1988, pp.253-261;

[3] V.Trajkovic, D.Davcev, G.Kimovski, Z. Petanceska, "Web - Based Virtual Classroom", In Proc. Of the TOOLS 34, Santa Barbara, California, USA, July 30 - August 4, 2000, pp.137-146;

[4] D.Rine, "Supporting Reuse with Object Technology", IEEE Computer, Vol. 30, No.10, October 1997, pp. 43-45;

[5] M.Balabanovic, "An Adaptive Web Page Recommendation Service", In Proc. of the 1st Int. Conf. on Autonomous Agents, Marina del Rey, CA USA, February 5-8, 1997, pp. 378-385;

[6] J.M.Corchado, B.Lees, N.Rees, "A Multi-agent System "Test Bed" For Evaluating Autonomous Agents", In Proc. of the 1st Int. Conf. on Autonomous Agents, Marina del Rey, CA USA, February 5-8, 1997, pp. 386-393;

[7] S. Hilts, B. Wellman, "Asynchronous Learning Networks as a Virtual Classroom", Communication of the ACM, vol.40, No.9, 1997, pp. 44-49;

[8] R.Chellappa et all, "An Electronic Infrastructure for a Virtual University", Communication of the ACM, Vol.40, No.9, 1997, pp. 56-58;

[9] M.F.Schwartz et al., A Comparison of Internet resources discovery approaches, Computer Systems 5, 4 (Fall), 1992, pp. 461-493;

[10] D.Hardy, et al., Customized Information Extraction as a Basis for Resource Discovery, ACM Transactions on Computer Systems, Vol.14, No. 2, May 1996, pp. 171-199;

[11] C.Bowman et al., Scalable Internet Resource Discovery: research Problem And Approaches, Communications of the ACM 37, August 1994, pp. 98-107;

[12] P. Baxendale, et al., An Introduction to Trilogy Virtual Laboratory, IEE 15th Teletraffic Symposium, March 1998