20

Pro-Active Multi-Agent System in Virtual Education

Victoriya Repka, Vyacheslav Grebenyuk and Katheryna Kliushnyk
Peoples-Uni,
Kharkiv National University of Radioelectronics
Australia,
Ukraine

1. Introduction

As virtual education becomes more and more widespread, its' application provides a unique opportunity for us to develop new applications in adaptive or intelligent agent technology. Adaptive or intelligent agent technologies allow education methods to be identified on a case-by-case basis, and undertaken regardless of location, time, age and life lifelong. There are many different distant education (DE) models developed to take into account modern tendencies of distributed system ideas and intelligent agent technology. The last strengthens the ontological features of DE system and move up the users from passive knowledge recipient role into active actors in the educational process. Therefore the main priority for the development of a modern virtual education system is to provide each student with an individual program, and to allow them to choose courses to fit their level of knowledge as well as make information searchable in accordance to the query criteria and within existing user skill sets. New methods and methodologies are being developed to solve this problem, as well as many others associated with virtual education, distant education, and e-learning.

An adaptation to a learner's personal interests, characteristics and goals is a key challenge in e-learning. In this chapter we discuss the architecture of web-based learning systems that addresses the learners' need for activities and content based on their preferences and equally considers the designer's and tutor's needs for the efficiency. The system aims to develop new methods and services for pro-active and adaptive e-learning. Proactive means the system involves acting in advance of a future situation, rather than just reacting. It means taking control and making things happen rather than just adjusting to a situation or waiting for something to happen. Adaptive means the learners are provided with a learning design that is adapted to their personal characteristics, interests and goals as well as the current context.

Currently, two approaches to adaptation are common within e-learning. In the first, dominated by a strong tradition in instructional design, a team produces a detailed design of content, interaction and presentation. Within the design different options may be worked out for different learners based on user data, e.g. level, interest or learning style. The options for adaptation are prepared at design time and require limited, if any, interaction of tutors at runtime. The second approach is based on the assumption that author and tutor is the same

person. The author designs the material. Next at runtime, the author, now as a tutor, adapts the course based on a direct interpretation of usage data, i.e. how well the learners succeed and what questions arise. However, both approaches tend to be (too) expensive because of high development costs or high delivery costs through extensive support.
Using of agent technologies in virtual education became very popular because of their properties like: reactivity – agents respond on environmental changes in the real time based on the rules of "WHEN event IF condition THEN action"; goal driven – agents have an ability to solve tasks and attain objectives, an ability to exist in a constantly active state having their own control flow; flexibility – agents' actions are not mounted rigidly; intellectuality – agents have an ability to learn, to find new solution, to change their behaviour with their own experience and experience of other agents as well. Due to these properties we can say the flexibility of action for such system becomes sufficiently notable and more over it has even some sort of "consciousness". We can individualize the virtual education process and build up individual educational trajectories for a student based on those agents' properties.

2. Related publications

Nowadays the agent technology has been applied in a various types of applications for education: to the design of peer-help environments, for information retrieval, for student information processing, distribution and feedback collection, pedagogical agents, teaching agents, tutoring agents, agents for assignment checking and agents for student group online support. Recently, agent e-learning systems are also used to perform mechanisms of interaction and collaboration in a network community environment.
In (Fuhua, 2005) it was suggested to call the adaptive learning environment for distributed learning as "a distributed adaptive learning environment." DALE (Distributed Adaptive Learning Environment) integrates agents and learning objects in the context of a distributed system. The main features of such environments are adaptability and distribution. They use intelligent software agents to provide the key intelligence to dynamically adapt the contents to the needs of particular learners. An empirical study aimed at the evaluation of the intelligent agents effectiveness in online instruction shown that agents can improve completion rates, learner satisfaction and motivation.
Samuel Pierre in (Samuel, 2007) operates an e-learning networked environment. This virtual environment can be defined as a software system within which multiple users, possibly located worldwide, interact with one another in real time (Singhal and Zyda, 1999).
Pashkin in (Burkhard, 2007) proposed agent-based architecture of intelligent distance learning system. This system contains agents divided into three groups: decision making agents that analyze data and require performance of actions from DLS side, Pedagogical Strategy Agent – interprets raw data received from Collector Agent and makes decision which rule should be executed, and Data Mining Agent -analyses of data to define association rules between student's behaviour and cluster students; interface functions agent – responsible for learning content presentation and human agents that model and represent of the human and perform tasks on behalf of them. The main goal is to model active interface for presentation of learning content depending on student's ability.
Interface or communication agents are available almost in all Multi-Agent Systems (MAS). For example in (Fuhua, 2005) they provide user support by interacting with the user. They help the user in completion of some task in an application. Interface agents carry out the user's instructions. In situations with no exact instructions to follow, agents act according to

the knowledge established through feedback from the user. Agents can also request help from other agents. Interface agents provide useful assistance to human beings. For example, e-mail filters learn to pick out junk e-mails by hints from the user and experience. According to (Fuhua, 2005) the MAS-based distributed-learning environment (DLEs), provide intelligent decision-making support as well as multitude of learning objects for students to choose and experience under various navigation environments. This environment infrastructure consists of the following agents: Information management agent; Intelligent decision-making support agent; MGIS support agent; Collaboration agent; and Interface agent.

The learning environments can support the interactions between agents that might be geographically dispersed on the Internet. DLEs provide an online training environment that enables students to acquire knowledge and improve safe navigation skills at any time and from anywhere, simply by using a Java-enabled browser.

Another similar environment, Intelligent Virtual Environments for Training (IVET) was proposed by Angelica de Antonio, Jaime Ramirez et al. in (Pechoucek et al., 2005). It is based on a collection of cooperative software agents. Among other things it includes agents able to simulate the behaviour of students and tutors, as well as agents able to plan the procedures to be taught prior to the tutoring process. IVET allows students to navigate through and interact with a virtual representation of a real environment in which they have to learn to carry out a certain task.

Adrianna Kozierkiewicz-Hetmanska established in (Nguyen et al., 2009) the concept for modification of learning scenarios in an Intelligent e-learning system. It is based on the ontology of knowledge structure of lessons. A set of linear relations between lessons takes into account. Such relation defines the order in which the lessons should be presented to a student, because some lessons should be learned before others. The system stores all student's preferences and changes in the user profile. Modification of a learning scenario is conducted in three steps. If student has a problem with passing a test for the first time he is offered repetition of the same lesson but in a different version. If the student fails again the system changes the lessons order, based on data of students who belongs to the same class. In another scenario if a student provides false information about himself then he may be classified incorrectly. If this possibility is taken into account, in the last instance the student is offered a modified lesson order based on all collected data.

All above works and also (Woda and Michalec, 2005), (Lin, 2007), (Shih and Hung, 2007), (Uhrmacher and Weyns, 2009), (Bellifemine et al., 2007) prompted our research.

Overall the intelligent learning systems are able to: conduct learner analysis based on initial interaction with the learner; adapt the instruction to meet the student learning style; monitor the learner progress, providing declarative knowledge when required; decide on the best way to present the next problem or instructional sequence; diagnose problems and provide corrective feedback; oversee the successful completion of the learning process.

In this Chapter the architecture and appropriate diagrams of the Pro-Active Multi-Agent System and the result of research the interaction of Jade agents on Jadex platform will be described within the virtual education purposes.

3. Pro-active approach to virtual education

Modern virtual education allows us to withdraw from strict academic program planning and lesson development in a virtual classroom. In order to improve virtual education and learning processes, the agent system, reacts to the users' actions and supports students in

choosing courses or modules of study, Software agents respond to user actions and build a personal trajectory of study modules. There must be a personal agent which controls all user's actions and registers them in the system.
The finite set of agents and message channel preside of student's flow through the learning module. The student has an access to the one learning object agent at the moment and follows to the instructions generated by agents according to beliefs, goals and plans. Under certain circumstances, an agent sends messages to another agent with different aims, e.g. to request information or to redirect the student to the next part of the educational program. The transitions and links between agents form the student's preferred educational trajectory. That helps the student to avoid a static modules sequence precompiled by a teacher. The coordinator agent controls the succession order of transitions. The main goal of the agent is to achieve the best subject coverage in a logical sequence and direct the student to meet course outcomes . The agent of a learning object communicates with the student and passes the contact with the student to other agent(s) to achieve appropriate learning objectives based on the student's knowledge level. In such way one can make nonlinear structure of learning content performance which is adapted to needs and achievements of the student. The basic classes of different e-learning content agents are considered in this Chapter. The basic structure for agent-to-agent and agent-to-student communications during the learning processes is researched as well.

4. Research of agent technology

As far back as in the nineties Pospelov published two fundamental works on the foundations of agents theory (Pospelov, 1997), (Pospelov, 1998). There he examined the historic transition from modeling of collective behavior to agents theory and introduced agents classification with the help of three criteria "type of environment - the level of 'free will' - the degree of social relations development," identified and analyzed the different types of agents operation environment, examined the key intentional agent characteristics. According to Pospelov prerequisites for the realization of a certain behavior for the artificial agent are special devices directly interpreting effect of the environment (receptors) and executive bodies that affect the environment (effectors), as well as processor (information processing unit) and memory. By memory we mean the agent ability to store information about its condition and state of the environment. Thus, the original idea of the simplest agent turns into the well-known "organism-environment" model (Pospelov, 1997).
Intelligent agent, generally, consists of the following components:

- interface - is responsible for agent communication with the environment and consists of sensors and effectors;
- agent database of knowledge - stores all the agent knowledge. It includes knowledge about the interfaces and other agents functionality, as well as current task data;
- scheduler - is responsible for planning further agent activities based on the existing knowledge about the task and environment.

The interface includes various functional units responsible for the interaction of agents and environment. Sensors are responsible for receiving messages by the agent from the environment and other agents (which in the case of the agent autonomy is equivalent to the environment by the manner of interaction).
Effectors, on the contrary, serve as the sender of agent messages to the environment and other agents. All interaction with the network neighborhood and other implementation

details are usually not included in the interface, as low-level tools (e.g. agent framework where the system is implemented) provide it.
The knowledge database serves as an agent for storage of all knowledge without exception, gained during the life cycle of the agent. This includes agent modeling database, the knowledge database about the tasks and knowledge base of their own "experience".
The knowledge database of the tasks contains statement of a problem, as well as the knowledge gained in the process of solution. It stores intermediate results of subtasks solutions. In addition, the database stores the knowledge about the task solutions and methods of selection of these solutions.
The knowledge database of the agent's own "experience" contains knowledge about the system, which can not be attributed to the foregoing categories. This knowledge database contains the solutions of previous problems and the various secondary (though, perhaps, useful) knowledge.
The scheduler is responsible for planning agent actions to solve the problem. As noted above, the scheduler must balance the agent activity between the plans construction for solving the problem in a changing environment and immediate implementation of plans (Pospelov, 1998).
All agents can be divided into five groups, according to the type of processing of perceived information:

1. agents with simple behavior - act only on the basis of current knowledge;
2. agents with the model-based behavior - can interact with the partially observable environment;
3. task-oriented agents - similar to the previous type, but they, among other things, store information about the desirable situations;
4. practical agents - recognize only the states when the goal was achieved, and when not achieved;
5. autonomous intelligent agents.

4.1 Agent platforms

An important concept of the multi-agent systems (MAS) implementation is the agent platform (AP) - system that allows you to create and delete, interpret, initiate and transfer agents (Weiss, 2000). Server environment contains agents in certain contexts (rights). For messaging intermediaries are used (to hide the code, have an opportunity to create their own copy or move to a different context).
Communication infrastructure (e.g., RPC) does for agent platform and its context for agent communication. The highest level is the region - set of AP.
Thus, the AP functions are the following: agent's communication, transfer of messages between agents of different platforms, support of ontologies, agent's management, search for agents and information about them within the system, management of agent life cycle, and security.
Platforms vary in range of application, technology (language, FIPA organization standards), design community, extensibility (API and plug-ins), integration with corporate systems, documentation, licenses, relations with business corporations and samples of projects.
The preferred platforms for development, standardized by FIPA are: JADE (most popular), Coguaar (no documentation), Aglobe (poorly supports FIPA), Jason (has own language AgentSpeak to describe the agents), Jack (commercial license).

For agent communication modeling MASON, RePast, Ascape, NetLogo are used. In our country, especially for academic and educational purposes, platform of multi-agent programming JADE (Java Agent Development Framework), fully implemented in Java using Java RMI, Java CORBA IDL, Java Serialization and Java Reflection API, is widely used.

JADE provides the agent system programmer-designer with the following tools:

1. FIPA-compliant agent platform - agent platform based on FIPA recommendations, including mandatory types of system agents: agent platform management (AMS - Agent Management Service), a channel of communication (ACC – Agent Communication Channel) and directory services (DF - Directory Facilitator). These three agent types are automatically activated when you start the platform;
2. distributed agent platform - distributed agent platform, which can use several computers (nodes), though each node runs only one virtual machine (Java Virtual Machine); agents are performed as a Java-threads. For agent messaging, depending on their location, the appropriate transport mechanism is used - Multiple Domains support - a number based on FIPA specifications DF agents that can be united to federation, implementing multiple domain agent environment;
3. multithreaded execution environment with two-level scheduling. Each JADE agent has its own thread of control, but also it is able to work in multithread mode;
4. library of interaction protocols - a library of interaction protocols using the standard interactive FIPA-request and FIPA-contract-net protocols. To create an agent that can operate according to the protocols the designer needs to implement only the specific domain activities; all the protocol logic, independent from the application, will be implemented by the JADE system;
5. GUI administrator - an administrative graphical user interfaces (GUI) provides simple platform management and shows the active agents and agent containers. Using the GUI, platform administrators can create, delete, interrupt, and resume agent activities, create domain hierarchy and multi agent DF federations.

JADE simplifies the process of multi-agent system development using FIPA specifications and a number of tools, which support debugging and system deployment. This agent platform can be distributed among computers with different operating systems and can be configured through the remote GUI. The platform configuration process is flexible enough: it can be changed even during the program execution; to do it, you need just to transfer agents from one machine to another. The only system requirement is installation of Java Run Time 1.2 on this machine. The communication architecture provides flexible and efficient messaging, while JADE creates the queue and manages the ACL-messages threads, which are private for each agent. Agents are able to access the queue using a combination of certain modes of their work: blocking, voting, shutdown, and pattern matching (concerning search methods).

Now Java RMI, event notification and IIOP are used in the system, but you can easily add other protocols. The ability to integrate SMTP, HTTP, and WAP is also provided for. Most of the communication protocols that have already been identified by international community of agent environments designers are available and can be illustrated by specific examples, after determining of the system behavior and its major states. SL (semantic language) and agent management ontology are also implemented as well as the support of user-defined content languages and ontologies that can be incorporated and registered by agents, and used by the system. For the purpose to substantially increase JADE efficiency there is a possibility of JESS and CLIPS Java-shell integration.

4.2 Jadex agent development technology

While carrying out the certification paper a number of software tools for the agent realization have been reviewed. Jadex represents a conservative approach to the agents. (Braubach et al., 2009) One of its advantages is that no one new programming language was used. Instead, Jadex agents can be programmed as object-oriented by means of such technologies as IDEs-Eclipse or Intellij IDEA, i.e. in Java. Jadex provides the basic structure and set of programming tools that facilitate agent creation and testing. In order to facilitate a smooth transition from traditional distributed systems to the development of multi-agent systems, there should be used, as far as possible, proven object-oriented concept and technology.

Research project Jadex is implemented by Distributed Systems and Information Systems Group at the University of Hamburg. (Pokahr, 2010) The developed program structure is currently under test. Primary function set is already capable to support the construction process of intelligent agents on FIPA-compliant JADE platform, and the system has already been put into use in some applications. Releases are available under GNUs LGPL license. Let us present some of the advanced Jadex features. Therefore, Jadex is Java-based FIPA-compliant agent environment, which allows, according to the "Belief-Desire-Intention" (BDI) model (it is shown on Fig 1) the development of target agents. Jadex provides the basic structure and set of development tools to facilitate agent creation and testing. Jadex project aims to maximally simple development of agent systems, which means its Java base. Using Jadex agent system can be created without a preliminary study of a new programming language. Jadex designed to facilitate the implementation of agents in widespread programming language Java, using a huge number of existing tools and libraries.

The aim of the Jadex research project is to create rational layer over the FIPA-compliant infrastructure that provides the possibility of constructing of agents technically based on audio software. To create an intelligent agent one needs to collect some components. Therefore, it is necessary to provide optimal agent architecture, which would take into account intra-agent concepts, agent society, and artificial intelligence. Specific character of this trend progress is that all the most interesting research results are in different fields that are isolated, independent and not joined into a well-organized architecture. Thus, until now, standards have not allowed us to create intelligent agents, considering all aspects. Jadex project provides open scientific information map that, firstly, outlines the fields of interest and, secondly, presents actual groundwork along with studies in progress. Due to the project openness anyone can contribute their ideas and practical improvements.

4.3 BDI architecture

Architecture agents require mental subsystems that help agents to communicate. Based on review the deliberative architecture was chosen, which had requirement functionality. One of deliberative architecture is BDI architecture.

In the development of models and agent architectures BDI architecture dominates (Pollack and Ringuette, 1990); the agent is considered there as a social creature, that communicates with other agents using a certain language, and plays a role in society, depending on beliefs, desires and intentions. It is considered that these three components fully define the state of social agent "intelligence". In terms of programming, BDI-agent beliefs are agent knowledge (information) about the state of the environment, which are updated after each action. Desires designate the goals of the agent, including their priorities. Intentions designate the actions to achieve the goal (behavior samples). Interaction protocols allow agents to reduce

the search area of possible solutions and identify a limited range of responses possible for this situation.

In hand-on programming agent is a shell structured computer system, situated in an environment and designed for flexible, autonomous actions in this environment in order to achieve desired goals. Agents alter from tradition software by complexity of interaction and communication scenarios. Ontologies are evident formal term specifications of application domain and the relations between them. This is term of social sciences, used in the theory of agents; it is almost equivalent to the programming concept of language semantics in the application domain.

Example of BDI architecture is presented in Fig.1

Fig. 1. BDI model

The most attractive BDI features are:

1. philosophical components that based on the theory of human rational action;
2. software architecture that was implemented and successfully used in several complex real applications. Examples: Intelligent Resource-bounded Machine Architecture (IRMA), Procedural Reasoning System (PRS);
3. logical components - the model is strictly formalized in the family of BDI logic (Wooldridge and Jennings, 1995). "Base BDI logic", introduced in Rao & Georgeff research (Rao and Georgeff, 1995) is a quantified expansion of expressive logic with time branch.

Intensions roles and features: intentions control mechanisms-outcomes analysis; intentions limit (invalid) reasoning / conclusion; intentions hold to the required direction; intentions affect the presentations, on which the prospective practical conclusion is based.

The agent accepts the obligation to some alternative, if the alternative overcomes the filter successfully and is selected by an agent as an intention. Obligations entail temporal stability of intentions. If the intent was accepted, it should not be immediately rejected. To what extent the agent is obliged to follow its intentions? "Blind commitment," "Single minded

commitment," "Open minded commitment." Agent obligations are relates both to outcomes, and to the mechanisms.

4.4 BDI usage in Jadex

"Belief-Desire-Intention" (BDI) - the means of thinking for intelligent agents. The term "intellectual tool" means that it can be used with various kinds of firmware that provide essential services to the agent of communication infrastructure type and management tools. The concepts of BDI-model, originally proposed by Bratmen as a philosophical model for the description of rational agents (Bratman, 1987) , have been adapted by Rao and Georgeff to some more models. They transformed it into a formal theory and model for software agent implementation, based on the concept of beliefs, goals and plans, but it is more suitable for multi-agent systems in software and architectural terms.

The notion of "agent", that is considered as a powerful example of software development, is very useful with regard to the complexity of modern software systems, including Jadex. It allows considering systems as independent interacting objects (entities, modules) that have their own objectives and act reasonably. Consequently, the internal state and the decision making process of agents are modeled intuitively, following the principle of intellectual relation. Focus means that (instead of direct call for agents for the implementation of any action) the developer can define goals that are more abstract for the agents and thus provide certain degree of flexibility to achieve them. BDI model, based on the intellectual relationship, originally was presented as a philosophical model for the creation of reasonable (human) agents. However, later it was adapted and transformed into a model for software agents, based on the ideology of the beliefs, goals and plans. Jadex collates this model into JADE agents by providing first-class objects with "beliefs", "goals" and "plans" that can be created inside the agent and which can be manipulated. (Pokahr et al., 2006)

Jadex agents have possibilities; they can be any type of Java object and be stored in the knowledge base. Goals are explicit or implied descriptions of states that are to be achieved. To achieve its goals, the agent fulfills the plans, which are Java coded procedural means.

Jadex structure consists of API, executed model, and predefined repeatedly used common functionality. API provides access to the Jadex concepts during programming plans. Plans are clear Java classes that expand special abstract class, which, in turn, provides useful methods of sending messages, organization of secondary objectives or events expectation. Plans are able to read and change the agent views, using the API belief base. In addition to Java coded plans the developer provides XML based agent definition file (ADF), which sets out the initial beliefs, goals and plans of the agent. Jadex execute mechanism reads this file, initiates the agent, and tracks its goals during the continuous selection and launch of plan based on internal events and messages from other agents. Jadex is equipped with some predefined functionalities - such as access to the directory facilitator service. The functionality, coded in separate plans, is grouped as reusable agent modules called abilities. Ability, described in a format similar to ADF, can be easily embedded into existing agents.

According to (Pokahr et al., 2006) BDI agent model in Jadex can be represented like this (see Fig. 2). These elements can be specified inside the ADF.

An important qualitative aspect of any development environment is the availability of tools support. For example, Jadex is JADE add-in, and thus a huge number of ready, available tools can certainly be used with Jadex. Nevertheless, this is true not only for tools that are available in JADE (such as Sniffer or Dummy Agent), but also for the third person instruments (such as bean generator plug-in for Protege). On the other hand, new notions

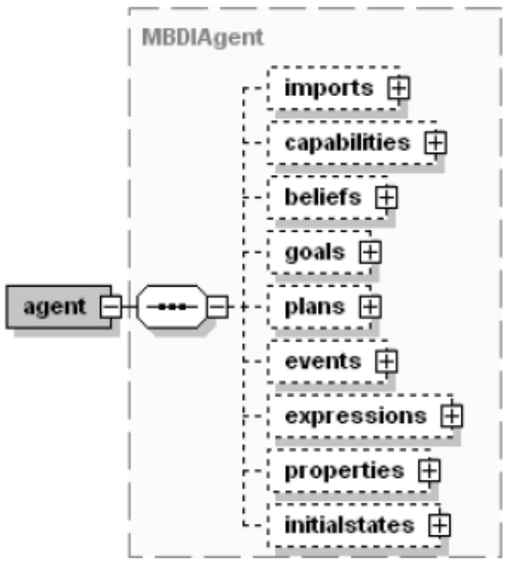


Fig. 2. Jadex agent XML schema

presented by Jadex, must also be supported. Considering the above reasons, the tools were implemented to assist working with these issues. BDI Viewer tool provides an opportunity to consider the internal state of Jadex agent that is its beliefs, goals and plans. The Jadex Introspector is similar to Jade Introspector, which allows to monitor and influence the agent behavior by means of observation and effect the way of managing the incoming events. Introspector is also capable to transfer agent to the step-by-step mode to eliminate errors. In addition to tools, Jadex has Logger Agent, which, if necessary, provides registered JADE and Jadex agents messages collection and demonstration, following the Java Logging API.

5. Multi-agent system for pro-active learning

5.1 Interaction schema

The Pro-active e-Learning Content Model focused on implementation of agent-oriented approach to developing computer-based learning systems. First of all, all learning object connect to appropriate agents. Complete set of the agents compounds learning content (LC) and with agents' messages forms network for a student to glide through it in studying of a subject.

At every instance the student has an access to only one agent of LOs (Learning Objects) and follows instructions generated by the agent due to its own beliefs, goals and plans. When some conditions would be reached the agent sends the messages to another agent(s) with different purposes, e.g. a query for help or to redirect the student to the next part of the learning content. Gliding from one agent to another, forms the students preferred learning trajectory. This enables the student to move away get from a static sequence pre-complied by an instructor. The sequence is controlled by LC Manager Agent. The main goal of the agent is to reach the best coverage of the LC and to assist the student in determining the optimal direction. If the student does take an unexpected direction, the LO's agents jumps in with help of a User Preferences Agent (UP agent). Therefore, in general the goals of LCM and UP agents can be mutually contradictory.

The general ideas of the Flow Control and Reconfigure processes are beyond of the scope of this paper. However, the role of them is very important for developing flexible and user friendly LMS.
Type of interaction between the user and LO agent depends from agents type. A skeleton of the interaction includes: request of the LO content; present the content under the specified conditions; student's feedback; assessment of the students' progress; determining of the next action. Some of these steps may be omitted or altered with concern to the specific of LO agent.
LO agents communicate with each other to send the information about current student's progress and in order to get an additional information and help. Most popular messages are on completion of the LO or reactivation of the some LO agent activity.
Every time when collisions happen in conversation between of LO agents, the LC Manager agent must resolve the problem and re-establish smooth flow of the educational process. So, its goal is not only to meet the educational requirements but also to maintain the order and consistency.

5.2 Types of Learning Object agents

By definition we know that there are many types of Learning Objects. Therefore, we concentrate our efforts on developing Agents only for theory, workshops, assignments, labs, and tests. Learning object may have in its structure some instructional content, glossary of terms, general metadata, quizzes, and assessments by the definition. So, all of this part can trigger different kind of user's action. However, not all of the LOs may include all of the listed elements. For example, simple theoretical notes may include only short portions of the instructional content and quiz with one or two questions, at the same time laboratory works can gather inside some theoretical text, instructions how to perform correct actions, initial data, pre-test, questionnaire, calculations, simulations, interactive models, and so on.
LO Agent for theoretical components of the learning content can be considered as a simple model-based reflex agent. That means it has very simple goals and reactive model of functioning based on condition-action rules. A user may perform some actions which generate conditions for agent's rules activation. Most complex cases of such agent functioning is when a cycle of delivering, showing, explaining, assessing repeats until the agent reaches a satisfactory user knowledge level. The agent can assess student's knowledge of their? content via simple single/multi choice questions.
All lecture notes, vocabulary articles, definitions, glossary terms, comments, online resources, learning films fall under this class of LO.
In addition to studying theoretical issues, discussions between students and teachers play very important role in every learning process. In discussion student may found an answer on questions or develop their own ideas. This process is different form interaction within theoretical components. Thus, another type of agent is needed to assist to that kind of learning activities. There are several triggered actions which can describe the work of users and agents on forums and workshops: Question, Answer, Opinion, Mark, Success, Fault, and Redirect. A sequence of these actions cannot be predefined. Therefore, there is no predefined way to fulfill the goal of such LO. The agent has to make plan and continuously to maintain it with re-planning. Such type of agents, let's call it Workshops LO Agent, could be implemented as utility-based planning agent.
Assignments are a type of the learning activities which stipulate that the student has to do some homework and represent his/her results on a teacher assessment. Generally, a teacher

may assign a task which is based on some theoretical LO or is an auxiliary to another LO. The teacher may also propose some script for fulfilling the assignment. The script states the dates, sample sequence, and list of links on readings. The student using all of the presented information completes the assignment and sends it for examination. Assignment LO Agent performs its action in some prescript sequence and may help student to find needed information, teacher to check posted assignments with known digital sources on coincidence, and both to be on the schedule. Thus, based on the analysis of agent's activities we can conclude it might be implemented as goal-based planning agent with library of preprogrammed plans.

During laboratory lessons students gain some practical skills and strengthen theoretical knowledge. With computer modeling we have got very powerful instruments for developing any kind laboratory setup for almost every subject. Labs LO Agent may shadows for student while he or she follows to labs instructions and assesses every single student's action. If help will be needed to student the agent may propose one of the stored standard instructions how to overcome the problem or send notification with request to the teacher. As well as the Assignment LO Agent, this type of agents is implemented in the same way.

Last type of learning activities is testing. We consider only tests with formal response like yes/no, single-choice, multiple-choice, and so on. An attractive feature of formal test is that they are particularly easy to score. Test LO Agent analyses student's results and find gaps in his/her knowledge. Agent's output depends not only from the student's results, but also from a type of the test: preliminary, self-test, monitoring, or final and grade tests. In any of these cases the agent has different type of actions which triggered when appropriate state is satisfied. We believe that this kind of agent can be successfully realized with simple model-based reflex agent.

Accordingly, we propose to develop Pro-Active e-Learning System as an organization of agents like:

- Simple model-based reflex agent for Theoretical and Tests Learning Objects
- Utility-based planning agent for Workshops Learning Objects
- Goal-based planning agent with library of preprogrammed plans for Assignments and Labs Learning Object.

5.3 Standard beliefs, goals, and plans for learning object agents

For defining an agent in Jadex, we need to formalize its beliefs, goals, and plans. Different types of agents listed above have different properties. Beliefs, goals, and plans also may differ from one LO to another. Here we describe most common ones for all types of LO agents.

Begin with goals. The most general goal for all Learning Object Agents is to fulfill a learning sequence and reach most good results with a student. Let's call it "Success" and prerequisites for it are completed goals "Sequence-Finished" and "Satisfactory-Mark". Both of these goals are achieve goals in term of Jadex. For checking of satisfaction of the mark level we add the fact "Current Student Mark" and "Mark Threshold" into agent's beliefs base. For the purposes of determining the level of the mark we also add "Mark Scale" class into agent's believe base. While "Satisfactory Mark" goal is achieved, the Jadex generates an event "Passed".

"Sequence Finished" becomes achieved when every single step of the LO plan has been done and an event "Done" is generated. An array of "LO Steps Completed" with boolean values is included into standard LO agent's beliefs base.

Some LO agent orients only on completion of the LO sequence, while others on reaching of satisfactory student's marks. We include "Priority" fact in belief base for dealing with importance of the goals.

A core goal for every single LO agent is "Show content" goal. This goal is a maintain goal type and must continuously to reestablish the "Show on Schedule" and "Student Will" conditions. Plans which maintained the goal use following facts of belief base: "Content.xml", "Glossary.xml", "Quizz.xml", "Sequence", and "FAQ.xml", and also can send event messages "Step Done: X" and "Mark is changed: +/-X".

Under some circumstances, achieving of a goal "Need Help" may be initiated. It can be doing by student itself or as a result of maintaining another goal – "Monitoring Student's Actions". Agent generates "Help" event message by using "List of Forced Errors.xml" and "Unforced Error" elements of belief base. "Need Help" goal's plans do a conversation with another LO agents in order to find a solution.

LO Agent during the whole lifecycle has to keep student's history. These data are stored into "Student's Behaviour History.xml" fact of belief base.

The last general goal for all types of LO agents is "Restart" goal. This goal operates with following facts: "Open Date", "Close Date", "Schedule", "Started Date/Time", "Restart from", "Student.xml". Achieving of the goal results in the agent drops all previous student's results and advancements (with exception of "Student's Behaviour History") and starts its function again (or from some point).

The "Restart goal" achieves at the beginning of LO Agent work or as a result of sending "Restart Forced by Student/Teacher" event message from "Monitoring Student's Actions" goal.

As output for LO Agent we define two types of ACL messages. First one is "Help Message.acl" and as it mentioned above it is used for establishing communications with another agents with the object to get an additional information to the student. Another one is "LO is Complete Message.acl". Agent sends this message when the "Success" goal is achieved (even with negative result).

For defining agent in Jadex we need to create XML agent definition file and Java classes for plans implementation. In Jadex if we want to define some abstract agent definition we need to use a capability definition. Some extraction from LO Agent capability definition is shown on the Figure 3.

Developing of the more specific types of LO agent has to be started with including a reference on the capability defined for the abstract LO agent.

5.4 Elements of multi-agent system for pro-active learning

Multi-agent systems are the system of different kind (software or hardware) in which a set of intellectual agents interact to each other, have some particular sets of goals and try to fulfill some sets of tasks. The finite set of agents and message channel preside of student's flow through the learning module. The student has access to the one learning object agent at one moment of time and follows to the instructions generated by agents according to her believes, goals and plans. Under some certain circumstances, an agent sends message to another agent with different aim, e.g. to request for an information or to readdress the student to the next part of learning educational program. The transitions and links between agents form the student's preferable educational trajectory. That helps to the student to avoid a static sequence precompiled by a teacher. The coordinator agent controls the

succession order of transitions. The main goal of the agent is to achieve the best covering of logical channel and to show the student on relatively optimal paths.
Interaction between user and agent depends on type of the agent. The skeleton of the interaction includes: a request of distant module content; its display to the student; an assessment of student progress; a determination of the next action. Some of these steps may be skipped or changed depending on the LO agent type. Most volume of messages concerns with educational séance completion or actuation of some particular agent's activity. Whenever collisions happen in communications between LO agents, the coordinator agent must resolve this problem and re-establish the smooth flow of educational process. It is so, because its goal is not only an execution of educational requirements but a support for order and sequence.

```
<capability xmlns="http://jadex.sourceforge.net/jadex"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://jadex.sourceforge.net/jadex
                http://jadex.sourceforge.net/jadex-0.94.xsd"
 abstract="true"
 package="jadex.planlib"
 name="Learning Object"
 package="ctde.jadex.paelc.lms">
 <beliefs>
  <belief name="current_student_mark" class="long">
   <fact>0.0</fact>
  </belief>
  <belief name="mark_threshold" class="long">
   <fact>60.0</fact>
  </belief>
  <belief name="mark_scale" class="long[]">
   <fact>{60.0, 66.0, 75.0, 90.0, 96.0}</fact>
  </belief>
 ...
 </beliefs>
 <goals>
 <!-- Restart LO Sequence. -->
  <achievegoal name="restart">
   <parameter name="open_date" class="Date">
    <bindingoptions>$beliefbase.openDate</bindingoptions>
   </parameter>
   ...
  </achievegoal>
 ...
 </goals>
 <plans>
 <!-- Send Help Message. -->
  <plan name="help_message_conversation">
   <body>new HelpConversationPlan()</body>
   <trigger> <goal ref="need_help"/> </trigger>
```

```
  </plan>
 ...
 </plans>
 <events>
 <!-- Student attended all LO Sequence. -->
  <internalevent name="done"/>
  <messageevent name="LO_is_complete" direction="send" type="fipa">
   <parameter name="performative" class="String" direction="fixed">
    <value>SFipa.INFORM</value>
   </parameter>
   <parameter name="language" class="String" direction="fixed">
    <value>SFipa.JAVA_XML</value>
   </parameter>
  ...
  </messageevent>
 ...
 </events>
</capability>
```

Fig. 3. Learning object agent's capability example

E-learning module may contain lectures, labs, tutorials, glossary items, tests, and some other type of resources in its structure. All of these types of LO can evoke different kind of user's actions.

A content agent (for example for a lecture LO) may be considered as a very simple one. That means he has very simple goals and model of quick functioning based on the learning rules. User can execute very simple actions which initialize conditions for firing agent's rules. The most complex case of the agent functioning is when the cycle of delivering, display, explanation, and assessment achieves of satisfactory level of the student's knowledge. The content agent can assess this student's knowledge via testing with or without a help of testing agent.

During laboratory works the students get some practical skills and improve their theoretical knowledge. So, the labs LO agent has to help to a student to do that. Every time when a student follows to the instructions of labs LO agent, the agent assesses his action. Whenever the student needs help, the content agents may provide one of the stored procedures which could help to overcome the unresolved problem or with the aid of the chat agent to send a message with a help request to a teacher. The student also may get an auxiliary help about interesting concept or phenomenon from context sources like glossaries and thesaurus by means of a context help agent even without involving of the teacher or tutor.

The resource agent is responsible for management of a search, selection and delivering of learning objects to the student. Its functioning is based on the operating with LO metadata which are compound into ontological model of learning resource.

The personal user agent is in action for establishing full-scale communication between the user and e-learning system. The basic task of the agent is maintaining a user profile with constantly incoming data (personal and educational).

Thus, for discipline and maintaining of process of active e-learning we are proposing to develop multi-agent system model which would include all of described above intelligent agents. A diagram of interaction between the agents is shown on the Figure 4.

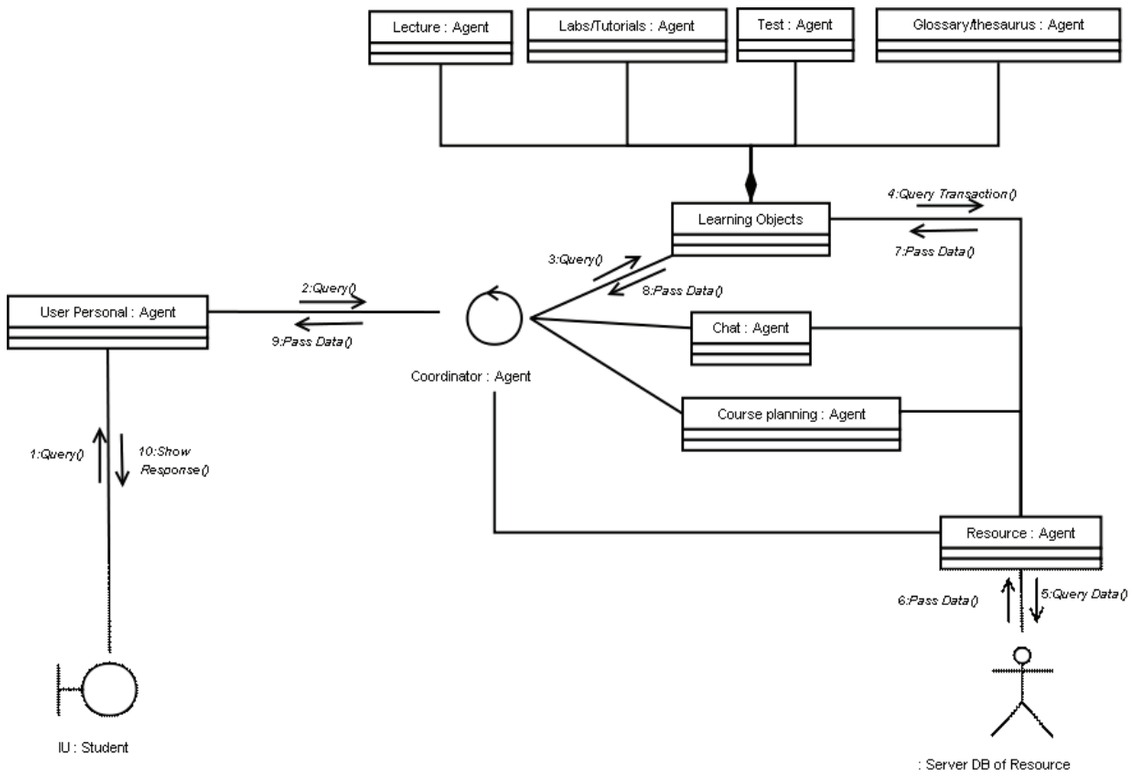


Fig. 4. Agents Collaboration Diagram for Multi-Agent Active e-Learning System

One can see the diagram of flow control sequence in proposed system on the Figure 5. The system contains following agent groups: 1. agents of communication: the coordinator agent, the resource agent, the chat agent, the user personal agent; 2. content agents: the lecture LO agent, the labs/tutorials LO agent, the test agent, the glossary/thesaurus agent. This set of agents is not comprehensive with listed kinds of agents and may be expanded. This expansion may be done on the basis of requirements analysis and need in enlargement of e-learning system functions.

As we mentioned above, we are supposing to use Jadex extension for JADE platform for developing the proposed system. Agents in Jadex are described with lists of believes, goals, and plans (Pokahr et al., 2006) .

All of listed agents have different properties and thus different believes, goals, and plans. For agent definition in Jadex one has to create two types of files: ADF file (Agent Definition File) – XML file with complete definition of agent and Java class files for execution of agent's plans.

Resources Agent. Allowing to obtain the search, selection and delivering of LOs to the student based on the ontological model. This model contains metadata elements for LO in accordance with LOM standard. Goals: main goal – data search (QueryInform) in metadata due to desired request from the coordinator agent, processing of query from the coordinator agent, transfer of discovered information (TransmitInform). The agent has five plans: Restart – resources agent restart plan; Search – requested data search plan; Message – plan for sending an error message to the coordinator agent in case of data not found; Transmit – result data transferring plan; Sleep – plan for sleep mode change-over.

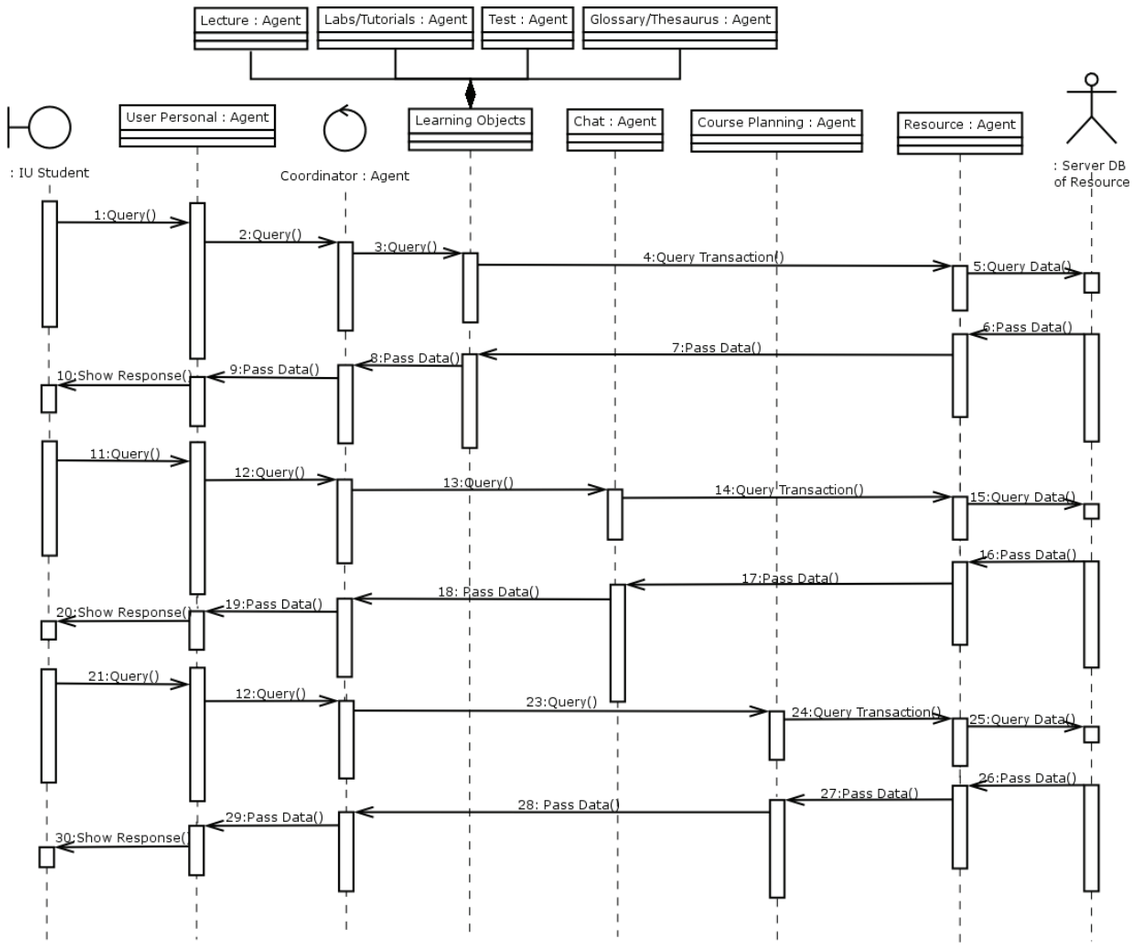


Fig. 5. Sequence Diagram

Agent's knowledge. Restart belief fact is setting true after the agent's start. QuerySearch fact shows a presence of data search query. InformNotFound represents a state after sending the error message to the coordinator agent when the search failure occurs. TransmitInform is a fact on success of information searching and of sending these data to the coordinator agent. SleepMode fact defines the moment when the agent moves to the sleep mode. That may happen if found information has been sent to the coordinator agent or the error message has been sent. Definition of the Resources Agent consists of two xml files: ResourceAgent.xml – agent's definition and Metadata.xml – e-Learning module metadata. The last one contains description of whole module content, locations of separate module elements, and short description of lectures, labs, tests, and other Los (Bratman, 1987) .

User Personal Agent. We introduce the user's personal agent the main goal of which is to maintain a user's profile and to support interaction of a user and the system. The agent collects all incoming information about the user and his activities. On the basis of these data it builds the user's model. Goals: authentication of a user; users questionnaire preparing based on the ontological model of the learner with the aim of getting personal preference; information sorting and structuring in accordance to the conceptual model of learner; educational goals tree forming, the tree consists of hierarchy of educational material subjects and corresponds to the subject area classification. Each user at the first moment has to be

registered in the system before starting his educational activity. That means he must obtain the login and password for system access and his personal account in a database has to be created. The agent provides an identification of the user on login and registers all his actions into the database.

The Course Planning agent. It executes the specific tasks directly related to the course planning by using agent-specific aspects, which details are given below.

System access is implemented through registration. There are two kinds of users in the system - an administrator and a student. Each of them has certain rights to access. Administrator can add or delete courses while student can choose courses or change selected courses if they are not started yet. The student can select available courses using a number of possible features: curriculum and student course; keywords; area of interests for the student (subject); course level (A, B, C).

Agents are active objects, which, unlike general (passive) objects they are not «sleeping » until they receive the next message (from user or environment) and perform it, but permanently operate solving the assigned tasks. Courses planning system can be supplemented by other agents, built on the principle of courses agent, such as coordinator agent or scheduling agent. Agents working in the system must ensure their proper functioning by adding resources of JADE-based agent technologies. The agent allows selecting for each student individually the courses for studying. The agent shows courses accessible to the concrete student (student interests, course level, and already completed courses are considered), allows to select several courses for learning (no more than four), it counts the credits for the selected courses. Also an available number of students on the selected course takes into consideration. After courses selecting the agent records it in individual sheet in which they are put in queue to learn. In this sheet it is possible to view when the course was started and whether it was finished.

Beliefs: User_Name; User_Password; User_Id; Max_Credits – default credit points equal 30 (basic knowledge). Goals: the main goal is to search courses based on student's preferences and education level.

Agent needs plans to work in an environment. With the help of certain expressions (or without its) the events are handled in the body of a plan; triggers, parameters, messages or purposes can be used here. Plans: Restart plan executes a new start of the agent after requesting from the coordinator agent; SignIn - extracts user data from database; Filter - allows user to choose different filters before showing available courses; SearchAvailableCourse displays a list of available course for concrete student taking into account user's filters; CheckCourses plan checks the correctness of selected courses (total number of credits, number of courses etc); AddCourses - allows adding courses to the student timetable; ListBadCourse plan allows user to exclude uninteresting courses from list of available courses; StartCourse plan looks at start and finish of each course for each student; CoursesList - displays list of all course (this plan is available only for administrator access); AddNewCourse - allows the administrator to add a new course to the courses list or edit data; DeleteCourse plan allows the administrator to delete course from courses list.

The Coordinator Agent. The agent is used for distribution of actions between agents and supporting them with necessary information. All information is stored in a database and marked up XML files on the e-learning server. An access to the files and database is accomplished by mean of inter-agent queries. Goals: Events monitoring; Selection of a module (course); Course material loading and analysis; LO agent initialization; Logging. Plans: display authorization form and get user information, and store it in beliefs set; load a

courses to appropriate belief; start the system and create the first event (Authorization, CourseInit), load a course content to appropriate belief; authorization; user's profile loading; get the list of course; display the course selection window; load the course; load the agent; get the resources; send a message; execute an application. Beliefs: User's profile, LO studying in progress, Current element of the course, Course structure, Course metadata files path.

The Glossary Agent. The main issue for the agent development is to support students with a context help. There are two main goals of this agent: Query and Main goal. The Query goal consists in displaying of information necessity, carrying out of condition search. The main goal is satisfied after displaying to the student all needed information. Beliefs: Restart agent fact; QuerySearch – search information query performance fact; ContentOpen – opening the content from a vocabulary fact; TermNotFound – a message on error would be sent to the coordinator agent; NoTerm – the fact about absence of necessary information and query to the teacher to enter the definition; SleepMode – the agent is in sleep mode fact. Plans: Restart, Reference, Search, ContentOpen, Coordinator, Chat, Student, and Sleep: the Restart plan execute a new start of the agent after a call from the coordinator agent; the Reference plan determines of hyperlinks existence in a text; the Search plan seeks for looking term; ContentOpen – displaying of vocabulary entry; Coordinator – sequence of communications with the coordinator agent. The agent performing this plan sends an error message on term not found. It is an external plan. Chat is an external plan for communication with the Chat Agent. It sends a message with info that there is no required information; Student – this plan display a message about incorrect link to the student; Sleep – after fulfilling this plan the agent falls into sleep mode.

The Lecture Agent. The agent supports the process of lecture notes reading by the student. Goals: Content displaying; Monitoring of the lecture window scrolling; Timing of the lecture reading; Putting testing questions; Getting a message about termination. Plans: Check if the student's name exists in the database, Load the lecture, Control of the reading time, Send a message to the helper, Display questions, Save the result. Beliefs: StudentEnrolment, ScrollPossition, ReadingTime, FinishButtonPushed, CheckFinalMark.

The Test Agent. The student may check the level of his knowledge after finishing some part of the educational material by means of the test agent. The agent may be summoned by the coordinator agent as well as any other LO agent directly. Goals: Run test on a chosen topic; Display test questions, Time control of testing; Calculate the score, Summarize the test result; Analysis result. Beliefs: Answer to a question is available; End of test; Test beginning; Screenshot Saving; Rating scale correspondence; Test time-out; Analysis of testing results. Plans: set up the subject at the test beginning; check whether the answer question is correct; assessing the mark; control of the test flow in depending on the test type; number of gathered points; time-out; the complexity of a question, and some another parameters.

The Labs Agent. The main issue of the agent is to maintain the process of laboratory works and tutorials carrying out by the student. Another goal is to give the students an opportunity to pass labs in their wished order with or without binding to the course content structure and sequence. Goals: to show labs content; to save the completed labs results; to send the report to the teacher. Beliefs: labs subject; labs sequence; labs tasks; an environment description; time for completion; acceptable idle time gap; labs report; address for sending the report. Plans: sending the subject for the coordinator agent; making the report for the labs; monitoring of the current time; checking of idleness time.

The Chat Agent. This agent is designed for supporting of the users' communication. The students may send some messages with their question to the teachers in case of needs and problems. Another useful application of the agent is to communication between the student and the agent as with a chat-bot, e.g. the student may ask some general questions or questions with formal answers. The chat agent consists of two modules: the communication module and the users' support module which works in the tutor absence. The communication module uses the text messages delivering service by establishing dual connection. By responding on requests the agent processes an event and makes a decision on establishing channel with the tutor. If the tutor is absent, then the agent passes control to the coordinator agent. The support module waits for receiving questions and then chooses an answer for them. The decision of the support modules is based on the ontological model of a dialog. This model includes concepts which are typical for questions and answers. The module compares request in question and templates in knowledge base and chooses an appropriate answers options. Then it checks the value for chosen answers and tries to find out which one is more applicable to the user's question. The agent allows to hold a conversation in user's language and to keep to main stages of common scenarios for conversation in e-Learning environment. Goals: Restart and ReplyBot. The first goal Restart is invoked when the request to the chat is initiated. The ReplyBot goal continues the function of the first goal and is applied when the user tries to connect to the tutor in off-line mode. Plans: For fulfilling the Restart goal the initialization plan is used. The plan shows the dialog window, loads the list of users by querying the coordinator agent and sets up a trigger for execution of chat-bot. A plan for achieving the ReplyBot goal is at finding of correct answer on asked question with help of ontological model. Triggers for query response are used for maintaining of conversation. In appropriate cases the plan uses agent's knowledge base and discovers what kind of answer the bot has to send to the user supplying with a virtual environment for users support.

6. Conclusion

The concept developing a Multi-Agent System for Pro-active Learning was promoted in (Grebenyuk and Repka, 2006). Since that the elements of this Multi-Agent System have been further developed and approbated. (Grebenyuk et al., 2006), (Repka et al., 2008) and others.
In proposed Multi-Agent System the intellectual agents interact with each other and allow the process of the active e-Learning. It means that the student is not anchored to a specific structure and sequence of the course learning, established by teacher. The student has an opportunity to learn the content in accordance to personal preferences, abilities, year level, and previous experience in using of distant education/learning systems.
The multi-agent system is able to self-learn from user responses and events during their course, and then determine and recommend help, thus, as much as possible simplify operations of the teacher and transfer this role from teacher to the tutor. In such a system the main role of the teacher is to prepare the qualitative learning material, and all teaching supporting processes will be conducted by intellectual agents.
To develop learning objects IEEE 1484.12.1-2002 Standard for Learning Object Metadata must be implemented. In the latest release of Multi-agent System one more agent was implemented – the Webbridge agent. (Kliushnyk, 2010) Webbridge enables agents to work through the browser, that standard Jadex features do not allow. The main goal within Webbridge is to support the communications of a user with the applied agents.

Webbridge allows combining the capabilities of agent technologies with the capabilities of web applications. Webbridge is a "glue tier" therefore it distinguishes between details of the planning agent and the web layer. By developing a system using Webbridge we can concentrate on developing agent-based business logic, and not on sending or receiving data to the browser. This allows you to develop a system on the base of MVC pattern, as Model 2 architecture. In Model 2 the data model is stored in data bases and the data are in Java beans for transmission and presentation. The accent in this architecture is fixed on the Controller, which is responsible for data transferring between the browser and agents.
Thus the proposed architecture of Pro-Active Multi-Agent System can be extended to any new types of agents in accordance with requirements of virtual education environment.

7. References

Bellifemine, F. L., Caire, G., & Greenwood, D. (2007). *Developing multi-agent systems with JADE*. Chichester, England ; Hoboken, NJ: John Wiley.

Bratman, M. (1987). *Intention, plans, and practical reason*. Cambridge, Mass.: Harvard University Press.

Braubach, L., Petta, P., Hoek, W., Pokahr, A., & SpringerLink (Online service). (2009). Multiagent System Technologies 7th German Conference, MATES 2009, Hamburg, Germany, September 9-11, 2009. Proceedings. Berlin, Heidelberg: Springer Berlin Heidelberg.

Burkhard, H.-D. (2007). *Multi-agent systems and applications V : 5th International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2007, Leipzig, Germany, September 25-27, 2007: proceedings*. Berlin, New York: Springer.

Fuhua, O., Lin (Ed.). (2005). *Designing distributed learning environments with intelligent software agents* Idea Group Inc

Grebenyuk, V., & Repka, V. (2006). *Pro-active E-Learning*. Paper presented at the International Conferences for Internet Technology and Secured Transactions, September 11-13, 2006, London, Great Britain.

Grebenyuk, V., Repka, V., & Shatovska, T. (2006). *Multi-Agent Systems for Providing a High Quality Distant Learning*. Paper presented at the International Conferences for Internet Technology and Secured Transactions, May 21-23, 2006, London, Great Britain.

Kliushnyk, K. (2010). *Research of model communication between JADE agents on the Jadex platform*. Linneus University, Växjö.

Lin, H. (2007). *Architectural design of multi-agent systems : technologies and techniques*. Hershey: Information Science Reference.

Nguyen, N. T. D. S., Kowalczyk, R., & Chen, S.-M. (2009). *Computational collective intelligence: semantic web, social networks and multiagent systems. first international conference, ICCCI 2009, Warsaw, Poland, October 5-7, 2009. proceedings* (1st ed. ed.). Berlin: Springer.

Pechoucek, M., Petta, P., & Varga, L. Z. (2005). *Multi-agent systems and applications IV : 4th International Central and Eastern European Conference on Multi-agent Systems, CEEMAS 2005, Budapest, Hungary, September 15-17, 2005: proceedings*. Berlin ; New York: Springer.

Pokahr, A. (2010). Jadex Software Projects.

Pokahr, A., Braubach, L., & Walczak, A. (2006). Jadex User Guide, Vol. 2010: Distributed Systems Group University of Hamburg, Germany.

Pollack, M. E., & Ringuette, M. (1990). *Introducing the Tileworld : experimentally evaluating agent architectures*. Menlo Park, Calif.: SRI International.

Pospelov, D. (1997). From collective of automata towards the multiagent systems. *The distributed artificial intelligence and multiagent systems*: 319-325.

Pospelov, D. (1998). Multiagent systems - today and the future. *Information technologies and computing systems*, Vol. 1: 14-21.

Rao, A., & Georgeff, M. (1995). BDI Agents: from theory to practice. *Proceedings of the First International Conference in Multi-Agents Systems*, San Francisco, USA.

Repka, V., Kliushnyk, K., & Kozopolyanska, A. (2008). The Agent Model of Testing Student Knowledge in Distant Learning System. *Bulletin of Kherson National Technical University*, Vol.1(30): pp. 417-421.

Samuel, P. (2007). *E-learning networked environments and architectures: a knowledge processing*: Springer.

Shih, T. K., & Hung, J. C. (2007). *Future directions in distance learning and communications technologies*. Hershey, PA: Information Science Pub.

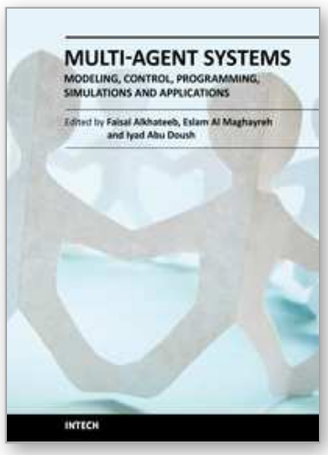
Singhal, S., & Zyda, M. (1999). *Networked Virtual Environments: Design and Implementation*. New York: ACM Press.

Uhrmacher, A., & Weyns, D. 2009. *Multi-agent systems : simulation and applications*. Boca Raton: CRC Press/Taylor & Francis.

Weiss, G. (2000). Multiagent systems a modern approach to distributed artificial intelligence: 648 p. Cambridge, Mass.: MIT Press.

Woda, M., & Michalec, P. (2005). Distance Learning System: Multi-Agent Approach. *Digital Information Management* 3(3): 198-201.

Wooldridge, M. J. E., & Jennings, N. E. (1995). *Intelligent agents : Workshop on agent theories, architectures, and languages : Papers*: Springer-Verlag.



Multi-Agent Systems - Modeling, Control, Programming, Simulations and Applications

Edited by Dr. Faisal Alkhateeb

ISBN 978-953-307-174-9
Hard cover, 522 pages
Publisher InTech
Published online 01, April, 2011
Published in print edition April, 2011

A multi-agent system (MAS) is a system composed of multiple interacting intelligent agents. Multi-agent systems can be used to solve problems which are difficult or impossible for an individual agent or monolithic system to solve. Agent systems are open and extensible systems that allow for the deployment of autonomous and proactive software components. Multi-agent systems have been brought up and used in several application domains.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Victoriya Repka, Vyacheslav Grebenyuk and Katheryna Kliushnyk (2011). Pro-Active Multi-Agent System in Virtual Education, Multi-Agent Systems - Modeling, Control, Programming, Simulations and Applications, Dr. Faisal Alkhateeb (Ed.), ISBN: 978-953-307-174-9, InTech, Available from: http://www.intechopen.com/books/multi-agent-systems-modeling-control-programming-simulations-and-applications/pro-active-multi-agent-system-in-virtual-education

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen