Research of model communication between JADE agents on the Jadex platform
Master Thesis

Research of model communication between JADE agents on the Jadex platform

Kateryna Kliushnyk

Supervisor: Ph D Anders Arpteg
Abstract

The relevance of the subject is caused by the increasing interest of society in agents and multi-agent technologies that are currently beginning to be applied in distance education. The purpose of the given paper is researching of agents communication model on Jadex platform.

The method of research is multi-agent systems and agent-oriented approach. For that express purpose the agent of course planning was constructed. It allows a student to choose courses for the certain educational program. This agent cooperates with JADE Webbridge agent which makes agents web oriented. They are constructed with the help of BDI model using deliberative architecture, i.e. in the work with agents black box sampling is used. Programming language Java is used, ADF files are XML-files.

The constructed program is to automate the process of course selection by students. Access to the course selection becomes easier and faster. Research of agents communication in multi-agent environment allows us to assert that proposed methods of agent implementation make possible to build a system of distance education, which is effective due to the proposed model of agents communication. The built system can be expanded and supplemented by new agents that will represent the certain objects of knowledge, such as lectures, practical lessons, etc.

Keywords: Intellectual agent, environment of agents, multi-agent system, architecture of agent, BDI architecture, Jadex, JADE, FIPA.
Acknowledgements

I would like to acknowledge my supervisor Victoria Repka, who encouraged, stimulated and helped me with this thesis. I am thankful to Anders Arpteg who gave me sound and initial ideas concerning the work. Especially, I would like to thank Tatiyana Shatovska, Mathias Hedenborg and Gosta Sundberg for the continuous support they provided me during my work on this thesis. I would like to thank developers of “Jadex” for the program libraries and tutorials. I also thank my family who supports me morally all the time, even if quite long distance separated us.
Table of contents

1 INTRODUCTION ............................................................................................................... 1
   1.1 URGENCY OF A TOPIC ......................................................................................... 1
   1.2 PROBLEM DEFINITION ...................................................................................... 1
   1.3 GOALS AND CRITERIA ....................................................................................... 2
   1.4 LIMITATIONS ....................................................................................................... 3
   1.5 OUTLINE OF THIS THESIS ................................................................................. 3

2 METHODOLOGICAL APPROACH ............................................................................... 5
   2.1 PROJECT LIFE CYCLE .......................................................................................... 5
   2.2 METHODS FOR BUILDING AGENT-ORIENTED SYSTEMS ................................ 6

3 AGENT-ORIENTED TECHNOLOGIES ....................................................................... 8
   3.1 SIMPLE INTELLIGENCE AGENT ........................................................................ 8
   3.2 COMPARATIVE ANALYSIS OF THE MAIN TYPES OF AGENT MODELS AND
      ARCHITECTURES ................................................................................................. 9
      3.2.1 Deliberative agents and architectures ......................................................... 10
      3.2.2 Reactive agents and architectures ............................................................... 10
      3.2.3 Hybrid agents and architecture .................................................................. 11
   3.3 AGENTS COMMUNICATION ............................................................................... 11
   3.4 AGENT PLATFORMS. JADE ............................................................................... 12
   3.5 JADEX AGENT DEVELOPMENT TECHNOLOGY .............................................. 13
   3.6 FIPA STANDARD ................................................................................................ 14
   3.7 BDI ARCHITECTURE ......................................................................................... 15
   3.8 BDI USAGE IN JADEX ....................................................................................... 17
   3.9 JADEX WEBBRIDGE AGENT ............................................................................ 18

4 COURSE PLANNING ..................................................................................................... 19
   4.1 UKRAINIAN SYSTEM FOR EDUCATION ........................................................... 19
   4.2 EUROPE SYSTEM COURSE PLANNING ............................................................ 21

5 DESCRIPTION OF THE SYSTEM’S RESULTS CONCEPTUAL, LOGICAL AND
   PHYSICAL DESIGN .................................................................................................... 23
   5.1 SHORT DESCRIPTION OF DEVELOPMENT ....................................................... 23
   5.2 ARCHITECTURE ................................................................................................. 23
   5.3 FUNCTIONAL SPECIFICATIONS DEFINITION .................................................... 24
   5.4 RESEARCH OF COMMUNICATION JADE AGENTS ON THE JADEX PLATFORM . 31
   5.5 THE DATA BASE STRUCTURE ......................................................................... 32
   5.6 SUMMARY .......................................................................................................... 35

6 SOFTWARE IMPLEMENTATION OF MODEL COMMUNICATION BETWEEN
   JADE AGENTS ON THE JADEX PLATFORM ................................................................... 36
   6.1 EQUIPMENT ENVIRONMENT .............................................................................. 36
   6.2 SOFTWARE ENVIRONMENT ................................................................................ 36
   6.3 COMMUNICATIONS REQUIREMENTS .................................................................. 36
      6.3.1 Communications overview ........................................................................ 36
      6.3.2 Communications software ......................................................................... 36
   6.4 INTERFACES ....................................................................................................... 37
      6.4.1 Modules of agent connection ...................................................................... 37
      6.4.2 Work with agents ...................................................................................... 37
7. CONCLUSIONS AND FUTURE CHALLENGES .....................................................43

7.1 RESULTS .............................................................................................................43
  7.1.1 Reflections .................................................................................................44
7.2 FUTURE CHALLENGES ..................................................................................44
7.3 CONCLUSIONS ...............................................................................................46

GLOSSARY .............................................................................................................47

REFERENCES .......................................................................................................48

APPENDICES .......................................................................................................51
  APPENDIX A ADF FILE OF AGENT OF COURSE PLANING .........................51
  APPENDIX B CODE OF SOME CLASSES...............................................................57
List of Figures

FIGURE 2.1: IMPLEMENTATION METHODOLOGY ................................................. 5
FIGURE 2.2: MUTUAL INFLUENCE OF AGENT-ORIENTED METHODOLOGIES ........... 6
FIGURE 3.1: THE MODEL SIMPLE AGENT................................................................. 8
FIGURE 3.2: FIPA AGENT MANAGEMENT REFERENCE MODEL ....................................... 14
FIGURE 3.3: BDI ARCHITECTURE................................................................................... 16
FIGURE 3.4: AGENT-BASED MODEL-2 ARCHITECTURE............................................... 18
FIGURE 5.1: COMMUNICATION JADE AGENTS ON THE JADEX PLATFORM IN BROWSER .................................................................................................................. 24
FIGURE 5.2: GOALS DIAGRAM ....................................................................................... 25
FIGURE 5.3: ROLES DIAGRAM....................................................................................... 26
FIGURE 5.4: USE CASE DIAGRAM ............................................................................... 27
FIGURE 5.5: SEQUENCE DIAGRAM ............................................................................... 29
FIGURE 5.6: ACTION DIAGRAM..................................................................................... 30
FIGURE 5.7: STATECHART DIAGRAM......................................................................... 30
FIGURE 5.8: FUNCTIONAL DIAGRAM, IDEF0, SECOND LEVEL DECOMPOSITION...... 32
FIGURE 5.9: ENHANCED ENTITY-RELATIONSHIP MODEL............................................ 34
FIGURE 6.1: JADEX WITH STARTED AGENTS.............................................................. 37
FIGURE 6.2: LOGIN PAGE.............................................................................................. 38
FIGURE 6.3: REGISTRATION PAGE................................................................................ 38
FIGURE 6.4: FILTERS PAGE.......................................................................................... 39
FIGURE 6.5: LIST AVAILABLE COURSES.................................................................... 39
FIGURE 6.6: INFORMATION ABOUT COURSE............................................................ 40
FIGURE 6.7: LIST UNINTERESTED COURSES.............................................................. 40
FIGURE 6.8: EXAMPLE MISTAKE................................................................................ 41
FIGURE 6.9: LIST INTERESTED COURSES................................................................. 41
FIGURE 6.10: START COURSE...................................................................................... 41
FIGURE 7.1: MULTI-AGENT SYSTEM OF DISTANCE LEARNING.................................. 45
List of Tables

TABLE 5.1: PRECEDENTS..............................................................................................................27
TABLE 5.2: DATABASE USERS AND AVAILABLE COURSES.....................................................32
TABLE 6.1: SYSTEM REQUIREMENTS.......................................................................................36
TABLE 6.2: COMMUNICATION SOFTWARE..............................................................................36
1 Introduction

Today, public interest in agents and multi-agent technology has increased. Agent technology includes the advantages of different disciplines. It includes the advantages of object-oriented and distributed software environments, the component-based concept of software development, and even artificial intelligence and expert systems. Agent technology can become a successful architectural solution, because it is able to integrate these different aspects and to reveal their inner potential. In this Chapter displayed main problems and goals system of course planning that based on agent oriented technologies.

1.1 Urgency of a topic
Dynamic development of new information and telecommunication technologies and their extensive use, as well as new groundworks in education and pedagogics allow to realize fundamentally new approaches to teaching. In the classical system of education there are a number of intractable problems that are the reason for constant search of new trends. At present, new methods of training are being worked out to enable people learn the material more effectively than by means of traditional teaching.

To solve problems that are associated with education, new methods and methodologies are being developed. One of the solutions is to use agent technology in which the system is constructed as a set of education objects, which are intelligent agents. The main objective of this thesis is to study the communication of JADE agents on Jadex platform, which due to their BDI architecture will be able to build a multi-agent system, effective because of the optimal model of agents communication.

To achieve this goal it is necessary to analyze the existing problems of distance education implementation and ways to resolve them. And also it is necessary to understand the modern methods and means of training organization, including agent technology, and to develop the agent of training subject, which can provide real support and assistance in the student's process of education.

1.2 Problem Definition
A future-oriented area is the study of agents communication. It will allow solving the problem of diversity of agent architectures and their types, since there is certain difficulty of agents selection for a particular software implementation, requiring considerable time and large outlay. Their variety allows agents to behave in different ways, which is the reason for increased system stability (reliability), but it is not always easy to predict their reaction to a certain event.

There are a number of problems associated with the agent technology standardization. IEEE LTSC, AICC, ADL, IMS, ARIADNE, and some other organizations took part in the various standards and specifications development. However, almost all the standards concern content interaction, data models and protocols. They do not cover all the components and multi-agent systems, such as the choice of architecture or the type of agent for a specific type of the task. The main part refers to the ability to achieve interoperability, accessibility and the possibility of repeated use via network of the education content by means of various types of metadata. It is important to note that currently there are no generally accepted standards for the creation of models of students in the systems of education, which is a significant deterrent to the creation of user-adaptive education systems.

At present, there is the paradigm of agent-oriented systems using intelligent agents as a high-level abstraction for the formalization and structuring of the subject area and as a powerful software tool for designing and implementing of complex information
systems. Agent allows monitoring the environment, designing, evaluating and choosing the ways of future actions, maximizing the expected efficiency of the education.

Large-scale implementation of modern technical means and technology into the education process has given impetus to the development of fully automated training systems, in which students would receive necessary educational material according to their wish and in the desired order. And also it is necessary to provide for the optimization of knowledge presentation to students, taking into account their personal psychophysiological qualities, level of training and knowledge. This means the opportunity to realize an integrated personality-oriented education model that takes into account the personal psychological special features of students, including the ability to analyze and correct their mistakes, made in the process of assigned tasks solution on the computer, as well as the opportunity to think about the nature of these errors.

The idea of our thesis is to realize this concept with the help of systems based on software and intelligent agents. The courses planning system, based on agent technology, solves the disadvantages of earlier courses planning systems, such as the lack of direct contact with the selection of certain courses.

Courses planning system has a number of undeniable advantages as compared to the traditional form of courses selection. The obligatory face-to-face meeting of a large number of students is not necessary. This is especially urgent for the large educational centers that have several branches geographically separated. An electronic selection of courses reduces the time for their organization. This allows meeting both personal educational demands and requests from educational organizations. One of the most important benefits of courses planning system is personalized approach to each student, which allows adjusting to their personal qualities, and therefore provides the most effective and efficient training.

The basis of this thesis is the construction of a model for courses planning system agent communication. Agents will qualitatively improve courses selection and planning. Especially it concerns the self-education systems that will adapt to each student, allowing to approximate to the real model of planning.

1.3 Goals and Criteria
This project is based on agent technology. The overall goal of the project is the construction of JADE agents on Jadex platform including all the inherent properties for the multi-agent system of the courses selection. For course planning system based on intelligent agent we should to define:

  1) the set of agents that have social behavior, reactivity, activity and base knowledge about the environment.
  2) the system of goals, facts and plans relations for each agent according to BDI architecture.

The main goal of this paper is to research the communication of JADE agents on Jadex platform, which will enable the construction of distance education system, it is effective due to the optimal model of agents communication. Depending on the parameters required for agents communication simulation we chose the two agents that are most closely interact together and which are sufficient to demonstrate the agents' work in the system of distance education:

  a) Webbridge agent, which is an interlink between web-fronts and agent of course planning.
  b) Agent of course planning allows the student to realize his own course planning.

The up-to-date distance education allows us to leave the strict academic planning process. In order to improve the distance education learning process we will establish the agent system, where the agents response the users' actions and depending on them
help him to choose his course. The purpose is the construction of course planning system of interacting JADE agents. In this paper we set following goals such as:

− to analyze the construction methods of agent-oriented systems and to choose the most suitable one.

− to analyze the existing agent architectures and to choose the most suitable one for the effective agents communication in the course planning system.

− to analyze the specific educational systems and course planning systems.

− to assign the agents required for the proper functioning of the course planning system and for system architecture constriction.

− the development of the course planning system model and agents communication model allows to construct a system that can simplify the course planning process for the students.

− to develop the structure of these agents that included goals, facts and plans (BDI schema) of agents, motivations and agent cooperate schemes. Agents are developed as agents of JADE architecture.

− the system should have the ability to be modified by existing objects expanding or new functions adding.

− to develop the user-friendly database, which should be easily transferred and used.

System of the agents should respond to user actions and analyzing them to build personal route of course planning. Using the abilities of intelligent agents, interacting with the external environment through the interface, the course planning agent enlarges the database, which then will turn into a network of its own knowledge. Following their own scheduler, the agents by means of hints should guide the student to the correct route of course planning.

1.4 Limitations

The technical solution of the project is to develop a course planning system on the basis of the intelligent agents. It works through a standard browser. The database structure is developed by means of Java – Hibernate abilities. The database is written as HSQLdb to port easy the data on different computers. The system was designed to research the agents’ communication on the Jadex platform, the main direction of development is to ensure the communication between agents.

1.5 Outline of this thesis

This thesis describes the research and development of course planning system based on intelligent agents. Chapter 2 introduces the basic concepts of the selected life cycle of program development and methodology for agent system developing. Chapter 3 describes the main concepts of agent-oriented approach, agent architectures, BDI agent
architecture, JADE environment and Jadex development platform. Here is also the brief description of Webbridge agent that was detailed in the system description in Chapter 5. Chapter 4 describes the course planning procedure and its introduction into educational system of Ukraine. Chapter 5 introduces the architecture of the system described using UML diagrams phase analysis and design, there is described the structure of the database and the BDI models was built for each agent and there was suggested the JADE agents communication model on the Jadex platform. Chapter 6 describes the technology used to implement the proposed software solution. We describe there the user communication with the system. Chapter 7 summarizes the main goals of the thesis and the results obtained on the basis of the work done in this thesis.
2 Methodological Approach

The selection methodological approach is an important step in development system. In the nineties of the XX century we witnessed a gradual increase number and capacity of methodologies of object-oriented programming. In the first decade of this century, this situation is repeating with the agent-oriented methodologies. However, unlike the object-oriented programming, the majority of methodologies yet do not have reached the stage of industrial application. And small researcher groups use it in the academic environment. Currently the majority of agent-oriented methodologies are used only for the establishment of small industrial systems. Firstly in this chapter, we selected project life cycle, then we selected method for building agent-oriented systems.

2.1 Project Life Cycle

B. Boehm describes it as follows: the main achievement of the spiral model is that it offers a range of opportunities for adaptation of successful aspects of existing models into the life cycle. At the same time, risk-oriented approach avoids many difficulties that are present in these models. In certain situations, the spiral model becomes equivalent to one of the existing models. In other cases, it provides the opportunity for the existing approaches to be joined to the best advantage in the context of this project (Boehm, 1988).

Spiral model has several advantages. It provides that every turn of the spiral corresponds to the creation of software fragment or version, it clarifies the purposes and characteristics of the project, determines its quality, and plans the next turn of the spiral. This model focuses on the early stages: analysis and design. At these stages the feasibility of technical solutions is verified by the creation of prototypes. Thus, the details of the project are deepened and consistently specified, and a valid version, which is to be brought to realization, is chosen. Evolution of a spiral model, therefore, relates to matters of work details. Especially we should pay attention to the refinement - requirements, design and code, i.e., giving greater importance to iterative, including increase of their numbers while reducing the duration of each iteration.

The life cycle of software according to RAD methodology consists of four phases: phase of requirements definition and analysis, phase of design, phase of implementation, and phase of deployment. Following them, we are developing our project. Initially, it is necessary to single out the features of our system and
requirements to it. Also we have to single out the necessary quantity of intelligent agents involved in solving the problems identified earlier. Then develop each intelligent agent separately and then join them into one full system. A detailed description of the development is in Chapter X, where models and functional principles of this system are given. While developing this system, we concluded that intelligent agents using is relevant in many areas that require the information accumulation, analysis, and processing. They are useful as part of the training to monitor level of knowledge, represent study material, practical exercises, building glossary, etc. Therefore, the aim of this project is to create an effective model of agents communication that can be applied in other areas of distance education as well.

2.2 Methods for building agent-oriented systems

Methods for designing agent-oriented systems are still in the initial stage of development (Shevcov, 2006). The known approaches can be divided into four main classes:
- those which are based on object-oriented methods and technologies using the relevant extensions;
- those which are used the conventional methods of knowledge engineering;
- those which have the organizational-oriented conceptions as their basis.
- those which combine the first three classes in various degrees.

Fig. 2.2 shows the mutual influence of the most currently known agent-oriented methodologies.

In the methodologies of the first class there are developing the expansion of object-oriented methodologies and technologies for the design of the agent-oriented systems. There are known the attempts of the direct applying of the UML-notation to represent the agent-oriented systems and patterns of agent communication. 

However, these proposals can not embrace the autonomy and proactive behavior of agents, as well as the variety of their communications. On the other hand, some suggest the extending and adapting of the object-oriented models and technologies to identify the methodologies for MAS. For example, it leads to extended models presenting the agent behavior and communication of agents and to the agent-oriented extensions of UML (Wooldridge, 2001). However, while these tools can provide the sufficient descriptions of the self-contained behavior of agents and their communications, they have no adequate conceptual mechanisms to work with organizations and communities of agents.

The second class of the agent-oriented methodology is based on the expansion of the conventional methods of knowledge engineering (Cetnarowicz, 1999; Schreiber et al,
These methodologies provide the formal and compositional modeling languages to verify the structure and functions of the system. They are very applicable for modeling of knowledge- and information-oriented agents. However, since these approaches usually require the centralized view on the systems, based on knowledge, they can not provide the adequate models and approaches for social consideration of MAS.

We chose the third class of methodology. These models and approaches are intended for the creation of the agent-oriented system from the «organizationally oriented» point of view (Ferber, 1998; Wooldridge, 2001). However, these approaches determine the organization just as a collection of interacting roles, thus not solving the problems of collective behavior of agents.

In the works of Wooldridge, 2000, Bordini et al, 2007 there was suggested and investigated the Gaia methodology. Gaia is the methodology of agent-oriented analysis and design that uses the organizational point of view. The most abstract entity in the hierarchy of Gaia concepts is a «system». Although the term «system» is used in the common meaning, it also means «community» or «organization». The Gaia methodology is investigated in the works of Mark Wood, Scott A. Deloach (2000). This methodology is called the Multi-agent Systems Engineering (MaSE). MaSE is the main goal of the methodologies for developing of the heterogeneous multi-agent systems. The goal of MaSE is to manage the developing of the system from the initial system specifications to the multi-agent system implementation.

The next level of hierarchy is the roles. The role is defined by three attributes: responsibilities, permissions, protocols. The phase of analysis tends to determine which the actual organization of agents community will be the best by means of decomposition of the system into the abstract «loci control», i.e. the roles played in the organization and ways of communication under the protocols. The expected outputs of the analysis phase: elaborated role model which defines each role in terms of responsibilities, protocols of communication and activities, as well as the communication models which describe each protocol in terms of data exchange and used samples.

The design phase includes the following stages: agent model identification, identification of services required for agents to perform the assigned roles, the development of knowledge model, the definition of design errors and, if necessary, back-off to the previous stages. Output of design phase: the model of agent system, which can be implemented using more conventional object-oriented and component technologies.
3 Agent-oriented technologies

Creating intelligence agent is a complex task that requires theoretical foundation. In this Chapter we describe main features of intelligence agent and creating IA on Jadex platform.

3.1 Simple intelligence agent

The term "agent" is derived from the Latin verb "agere", that means "to act", "to move", "to rule", and "to manage". Here is the definition taken from the encyclopedic dictionary of Brockhaus F.A. and Efron I.A.: "agent is a figure or a person acting on behalf or by authority of the other person." This definition rightly expresses the essence of modern intelligent agents (IA), which can operate autonomously on behalf of its owner (human user or other computer system) and to solve various tasks of information processing. To fulfill its work successfully, the agent must have sufficient intellectual ability to interact with the owner to get tasks and transfer the results, to be guided by its environment and take the necessary decisions (Pospelov D.A., 1997).

Since the generally accepted definition of intelligent agent has not yet been developed, one should speak about it as the class of agent objects (entities), including many species. We will consider the IA to be a software or hardware object (entity), operating autonomously to achieve the goals set by an operator or a user and to have certain intellectual abilities. The level of the abilities required for the goal attainment by IA, can be determined using the classification of Pospelov D.A. (Pospelov D.A., 1997).

As far back as in the nineties Pospelov D.A. published two fundamental works on the foundations of agents theory. 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 D.A., 1997) (Fig. 3.1).

Figure 3.1: The model simple agent
Intelligent agent, as a rule, 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. Sensor is 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). Effector, on the contrary, serves 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. Also, the database stores the knowledge about the task solutions and methods of selection of these solutions.

The knowledge database of the agent 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.

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 D.A., 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;
2) task-oriented agents - similar to the previous type, but they, among other things, store information about the desirable situations;
3) practical agents - recognize only the states when the goal was achieved, and when not achieved;
4) autonomous intelligent agents.

3.2 Comparative Analysis of the Main Types of Agent Models and Architectures

Creating IA is a complex task that requires theoretical foundation for the conceptual representations of agents (Gaaze-Rapoport, 1987). IA model serves as such foundation that in different ways describes the knowledge, ways of reasoning, planning of the behavior and direct actions of agents.

These models should be considered in two aspects: first, from the point of view of analyzing the properties and behavior of agents in MAS operation as a whole, and second, from the point of view of the study and design of the agent properties that determine its internal processes (knowledge acquisition, goals development, decisions making, etc.).
Initially (1956-1985), a lot of agents, developed in artificial intelligence, were agents, performing symbolic inference. The problems of organization of a symbolic inference have lead to rejection of such architecture, and lead to the different type of architecture - the concept of reactive agents, since 1985 to nowadays. Since 1990 to nowadays a number of alternatives were suggested: hybrid architectures (Shevcov, 2006).

We can take as a fixed notion the tradition to single out three basic classes of agent systems architecture and their appropriate models of intelligent agents: deliberative architectures and models (deliberative architectures); reactive architectures and models (reactive architectures); hybrid architectures and models (hybrid architectures).

3.2.1 Deliberative Agents and Architectures

Architecture, or agents, which use only exact picture of the world in symbolic form, and solutions (e.g., actions) based on formal reasoning and the use of comparison methods with the sample are determined as deliberative. As examples of this approach implementation can serve the following systems: Integrated Planning, Execution and Monitoring (IPEM), Pheonix, Homer, Grate, and others.

The organization of MAS as a deliberative architecture has an advantage from a perspective of symbol knowledge representation convenience. But at the same time creating an accurate and complete model of the world representation, processes and mechanisms for reasoning have certain difficulties.

Deliberative architecture is acceptable as agent architecture, containing accurate symbolic model of the world and making a decision on the basis of the inference. Deliberative agent has to possess the following features:
- to contain the explicitly expressed knowledge database, that represents its beliefs, loaded with formulate in a certain logic language;
- to function in a following cycle: environment perception (observation) - inference - action...;
- to make decision about the actions on basis of an inference.

The advantage of deliberative models and architectures is the possibility to apply rigorous formal methods and well-tested technologies of traditional AI that allow knowledge to be relatively easy represented in symbolic form and transferred to the agent-oriented systems (Shevcov, 2006). At the same time, the creation of complete and accurate model of some application domain of the real world, the formalization of mental properties of agents and processes of reasoning in these cognitive structures are significant challenges for their technical implementation.

3.2.2 Reactive Agents and Architectures

Search for problem solutions during the use of classical methods of AI in agent systems led to the new class origination - reactive architectures. R. Brooks is considered to be the founder of this trend (R. Brooks, 1986); he who formulated key concepts of behavioristic view of the intellect: intelligent behavior can be created without the explicit symbolic knowledge representation, intelligent behavior can be created without the explicit abstract logical inference, the intellect is emerging feature of some complex systems.

Reactive agents do not have any internal symbolic model of the world and work according to the rule type "situation - action", choosing the most appropriate actions to meet a specific situation. These agents are simple in structure and relatively well-adapted to work online, almost in real-time mode. One of the disadvantages of reactive agents is difficulty to implement task-oriented agents. Reasoning of intellectual agents in this case is common; they are not flexible enough and are too slow to respond the external events. The classic example of reactive architecture is Brooks architecture (subsumption architecture).
3.2.3 Hybrid Agents and Architecture
Reactive approach allows efficient use of the set of simple scenarios of agents behavior within the determined reactions to certain events of the environment, but its boundedness becomes apparent as the practical impossibility of full situational analysis of every possible agent activity. Therefore, in most projects and existing systems hybrid architecture (Shevnev, 2006) is used.

Recently, several researchers have recognized that the IA should have a high level output and low-level reactive capabilities. Thereby, we can use the reactive capacity for current tasks and add the possibility of inference for more complicated long-term tasks.

Long enough two categories of hybrid agent architectures were singled out. Homogeneous architecture uses one common representation and control scheme for the reactions and reasoning, while the multilayer architectures use different representations and algorithms (implemented in the individual layers) to perform these functions.

Reactive component displays perceptual stimulus for primitive actions. Deliberative component is a symbolic conclusion to control the reactive components behavior, for example, changes the set of “situation - action” rules. Deliberative component in some architectures is directly connected with agent sensors and effectors, while in others is not. Today, hybrid architectures are known in terms of the items mentioned above, such as: touring machines architecture, "planner-reactor" agent architecture, Dyna architecture, Intrrap, Oasis, Circa, and other systems.

3.3 Agents Communication
There are three common types of interaction between agents (Nijholt, 2004) that does not break their compatibility: the remote creation of an agent, agent relocation, and agent methods activation.

It should be noted that only agent relocation is directly related to mobile agents.

When remotely creating an agent, client program interacts with the required agent system to create an agent of a specific class. We can use a client program:
- non-agent system program;
- agent of an agent system with the type different from the agent system, in which the agent is created;
- agent of an agent system with the same type as the agent system, in which the agent is created.

Client is authorized in the agent system, and the created agent obtains the privileges of the client. The client sets the initialization arguments and, if necessary, creates the class object and initiates the agent. Agent system can also build agents on its own initiative. New agents usually have the same authentication as the agent system.

Agent relocation. When the agent is transferred to another system, the agent system creates a relocation request. Part of the request for relocation is the name and address, which determine the transfer target. The agent also determines the quality of communication needed for agent relocation. If the source agent system reaches the transfer target (agent system), it must perform the relocation request, otherwise it must return a message about the impossibility of relocation to the agent.

When the target agent system confirms the relocation, then status, authentication, and security settings are transferred to this agent system. After the transfer is over, this agent system initiates the agent and continues its performance.

Agent methods activation. The agent activates another agent or object methods, if it is authorized and has links to the object. As in the case of the transfer, agent determines the necessary level of quality for the service. Communication infrastructure must perform the appropriate method and return the performance result, or to report the performance error. When the agent performs the method, this performance uses the privileges and permissions of the performing agent.
3.4 Agent Platforms. JADE

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. 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 to 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: agents communication, transfer of messages between agents of different platforms, support of ontologies, agents 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 platform for development, standardized by FIPA (section 3.6), are: JADE (most popular), Coguar (no documentation), Aglobe (poorly supports FIPA), Jason (has own language AgentSpeak to describe the agents), Jack (commercial license).

For agent communication modeling are used: MASON, RePast, Ascape, NetLogo. 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) administration GUI - graphical administrator interface providing simple platform management and showing 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 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 the system. For the purpose to substantially increase JADE efficiency there is a possibility of JESS and CLIPS Java-shell integration.

JADE is used by a number of companies and academic groups. There are a number of well-known among them: BT, CNET, NHK, Imperial College, IRST, KPN, University of Helsinki, INRIA, ATOS, and many others.

3.5 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 (Alexander Pokahr and Lars Braubach, 2007). 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, ie in Java. Jadex provides the basic structure and set of programming tools that facilitate agents 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 concepts and technology.

Research project Jadex is implemented by Distributed Systems and Information Systems Group at the University of Hamburg. 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. So, Jadex is Java-based FIPA-compliant agent environment, which allows, according the BDI model, 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, that 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.

3.6 FIPA Standard

Foundation for Intelligent Physical Agents (FIPA) develops approaches to the definition of semantics for information on the basis of interoperability, as well as software standard for heterogeneous and interacting agents and agent-oriented systems, including extensive specifications that regulate intelligent agent communication, their structure and program implementation features. FIPA's approach to MAS development is based on creation of a framework for agent management in an open environment (Fig. 3.2).

This framework is described with the help of the model of a standard environment, in which agents are created and operated, and agent platform, which defines the infrastructure for agent deployment and interaction.

![Figure 3.2: FIPA Agent Management Reference Model](image)

In FIPA agents interact by means of messaging, that is an action encoded in ACL (Agent Communication Language). One of the key roles in the system is played by the services. They are required to provide various services to agents (e.g., message transport service). In this case specific service realization is of no importance: it can be implemented both in the agent form, and in the software form, that can be initiated by the previously designed interfaces. However, agents that represent services are more limited than regular agents, as they have to keep the service semantics and do not have enough autonomy (they can not make decisions to deny a service).

Primarily, the agent obtains access to basic services necessary for its existence, such as message transport service (message-transportservice), agent directory service (agent-directory-service) and service directory service (service-directory-service).

One of the main Jadex technology success factors is well-timed availability of standards that guarantees interaction possibility for the developing products. To facilitate the interaction of independently developed multi-agent systems, FIPA issued a set of requirements, which are usually referred to FIPA-standards. FIPA standard exactly defines agent platform architecture - agent controls and directory facilitator. This architecture allows agents to communicate, using the usual language of agent communication. In order to achieve FIPA-compatibility, Jadex was based on the JADE Agent Framework - an open development source, created by Telecom Italia Lab. JADE provides the platform architecture, the basic services and the mechanisms of communication required by FIPA conditions.

In the abstract FIPA architecture agents interact by exchanging messages which represent speech acts encoded in the language of their communication; service providing services for agents (including directory services and messaging) and can be implement-
ed either as agents or as called software that can be accessed using a software interface (for example, in Java, C++, or IDL). Thus, it is possible to identify information exchange between agents and access to directories as information formats, recognized at the infrastructure level.

**Agent Directory.** Function of agent directory is to provide information about agent registration. The agent directory entry is a pair of values, the first is agent unique global name, and the second is one or more structures of transport descriptions required for this agent access: description of the type of transport, the specific transport addresses, and if necessary, additional features required for communication with the agent.

In addition, agent directory service may contain other descriptive characteristics, such as services offered by the agent, expenses associated with agent use, agent use limitations, etc.

Agents can use the directory service agents to search for other agents they must communicate with. Technically, this means the search for the required agent directory entry. If the search is successful, the agent receives an entry about the required agent. The agent can also obtain other agent directory entries that meet search conditions. Then the agent can return to the study of agent directory records to identify an agent that meets its needs to the best advantage.

**Service Directory.** The main task of the directory services (service-directory-service) is to provide agents and services with facilities to retrieve other services. Service directory provides space for service registration by creating service directory entries (service-directory-entries), containing descriptions of these services. In addition, agents and service use service directory to find services that meet their needs.

Nominally, service directory is similar to the agents directory; the only difference is that the latter is necessary for agent search, and service directory - for service search. However, in practice, their implementation is very different: agent directory is often represented as a small size fixed table, while the service directory for its implementation uses more complicated distributed technology.

**Agent messaging.** The main function of FIPA-system agents is to communicate with each other. Here are some of the basic concepts of agents and their relations, which enable to organize such a dialogue. Each agent has a name. This name is a unique and permanent. Each agent has one or more transport descriptions, which are used to send messages to other agents. Each specific transport description is associated with one of the appropriate forms of transport message, such as SMTP or HTTP protocol. Transport is communication mechanism. Transport message is the message encoded for transportation in the required format, depending on the type of transport used. Transport description can affect the agent locator. For example, an agent named "TestAgent", which can be accessed by two transports (e.g., HTTP and SMTP), will have two transport descriptions used in the agent locator: for HTTP transmission a specific transport address is http://test.org/testagent, and for SMTP - testagent@test.org. Another agent may communicate with "TestAgent", using any transport description, and to find out from this description the agent that interacts with it.

### 3.7 BDI architecture

Main requirement to architecture agents of system of course planning is availability mental subsystem that helps agents in communication. On base review chose the deliberative architecture, which had requirement functionality. One of deliberative architecture is BDI architecture.

In the development of models and agent architectures BDI architecture dominates (Bratman, 1988); 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 agent to reduce the search area of possible solutions, 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. 3.3

Figure 3.3: BDI Architecture

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 (Rao & Georgeff (1992), Wooldrige (1995)). "Base BDI logic", introduced in Rao & Georgeff research 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.
3.8 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, 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. Systems, developed in these frameworks, are called procedural reasoning systems (PRS) as regards to their first representative.

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.

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.

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 3rd person instruments (such as bean generator plug-in for Protege). On the other hand, new notions 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.

3.9 Jadex Webbridge Agent
CMA trend appeared at the junction of different trends, such as artificial intelligence, parallel programming, Internet communications, and now is rapidly developing (Alexander Pokahr and Lars Braubach, 2008). The main element of system software agent that enables it to make decisions, plan activities, interact with other agents is the ontological knowledge base containing a model of conceptual notions, application domain relations, and rules for analysis and situational orientation.

Today, web-based applications play a leading role in the computer industry. The access is implemented through standard web browsers, which is very convenient for any user. A lot of frameworks are being developed to facilitate the development of web applications. Jadex environment does not work with web applications, but it became possible due to the student's research. Jadex-web-bridge agent made possible the development of web agents, using the standard possibility of Java. We can see agent performance algorithm in the model (Fig. 3.4).

![Figure. 3.4: Agent-based Model-2 architecture](image)

The agent works with BDI architecture, and is controlled by FIPA protocols. Agent inherits the architecture of Model-View-Controller. It is a bridge between the agent and a web browser. To develop web agents with use of Jadex Webbridge a standard set of files similar to Jadex agent is used, according to the IDE standard.

As you can see from Figure 3.2, the browser sends a request to web-bridge agent, which refers, according its goals, to plans of the agent, and displays necessary jsp page. The data, which will be transferred from this page, will be processed first by this agent and then transferred to another agent in the form of a standard message. The specific character of the creation of an agent that performs via Jadex Webbridge is as follows: classes of agent-bridge are originally specified in web.xml file. They turn all incoming requests into the messages and transmit them to the agent that sent the request, for further processing.
4 Course Planning

The process of creation of single Europe, dictated by economic suitability and increased society mobility, is inseparably connected with the development of European cooperation in higher education. This Chapter describes different systems of education and their differences.

4.1 Ukrainian system for education

The role of knowledge needed to consolidate its intellectual, cultural, social, scientific and technical capacity is constantly increasing today. Efficiency and quality of acquiring and dissemination of knowledge are determined by the level of development and the proximity of national educational systems, including higher education.

Higher education in Ukraine is of high quality. During 18 years of country independence, a well-developed, competitive and diversified system of public education was built in the educational space of Europe. Ukraine entered the III millennium to advance its own educational policies and strategies. Its basic principle of the education system corresponds to highly developed countries.

Now system of standards for each level of educational qualification and type of study is legally approved in Ukraine. The standards developed for 80% of areas (qualifications) contain all the requirements for the competence, qualifications, characteristics and knowledge quality determination system.

Currently, Ukraine has more than 800 higher educational institutions of all accreditation levels and forms of ownership: universities, academies, institutes, colleges, technical schools and specialized schools.

The structure of higher education in Ukraine is based on the educational system of the developed countries of the world, in accordance with the recommendations of the UNESCO, the UN and other international organizations.

Higher education is an integral part of Ukrainian education, as stipulated in the Law of Ukraine "On Education". Four-level system provides a thorough academic, professional and practical training for the next level: Junior Specialist, Bachelor, Specialist, Master.

There is no system of continuous higher education in Ukraine. As in most post-Soviet states, science (especially the fundamental one) is traditionally the prerogative of the National Academies of Science, distanced from the educational process in universities. The same can be said about the training of highly qualified specialists - candidates and doctors of science.

Higher education can be obtained in higher education institutions of certain accreditation level. Candidates should have a basic general secondary education, senior secondary education, or have Junior Specialist or Bachelor's degree, or Specialist or Master degree, if they apply for postgraduate course.

The multilayer structure of higher education system means that upon completion of education at this level, students receive an appropriate degree and can move up from level to level.

Students can take either full-time tuition (day), part-time courses (evening, distance learning), or their combination. Sometimes they can do an external course. Now distance learning becomes popular as well.

The academic year in Ukrainian higher education institutions begins on September 1. It is divided into two parts - the semesters, the first in the period from September to the end of January and the second begins in February and ends in June. Each semester lasts for 17-18 weeks, then 3 weeks are devoted to the examinations.
As a member of Bologna process, Ukraine intends to join the common European education and European academic research community by 2010. Thus it will participate in the creation of "Europe of Knowledge", and become more competitive.

Bologna process aims at increasing the international prestige and competitiveness of European higher education system by creating a single educational space, the main features of which are the follows:

- The quality of higher education, including quality of content, soundness, and the universalization of the conditions of educational programs realization, as well as the level of professional qualification of graduates;
- Mobility of students, teachers, researchers in a single educational space and free relocation of graduates with their employment;
- Diversity and flexibility of content and realization technology of educational with regards to traditions, autonomy and academic freedom of European universities, openness and accessibility of education, ensured by mobility, mutual opening of university branches in different countries and the introduction of distance learning technology and organizational structure.

The problem of diploma recognition in Europe is solved with the help of credit system, developed on basis of European Union document ECTS (European Credit Transfer System), established under the ERASMUS program and approved by the number of academic institutions. This system allows comparing and mutually converting the university academic load. Training courses are evaluated by conditional units (credits) and reflect the level of students academic load (lectures, seminars, self-direct learning, practice, etc.). Each course has certain number of credits that student gets after passing the exams. The popularity of ECTS grows; more than thousand educational institutions of the EU countries use this system today.

European universities successfully employ the system developed by the ECTS. To obtain master degree a student of the International Master Program must study at least 6 courses per year, 3 courses per semester. Requirements for courses combination may vary with the appearance of additional courses in economics, ethnology, and art history. But it is not actual for Ukrainian education.

Studying of subjects and disciplines is determined by the state educational standard (basic vocational education program) of higher vocational education, approved by the federal authority, curriculum, approved by the Academic Council of the university, and the program approved by the rector of the institute of higher education.

The central document defining the content of general vocational and professional (special) training is a training plan. It implements the basic principles of subjects selection, their systematization, regulates the volume of academic disciplines, student load for periods of training, types of knowledge control.

All subjects included in the plan are linked, that is, in more recent subjects information from previously studied subjects is used without its specificity, i.e. assumed that the student knows what particular definition or concept means. For more detailed examination of the structure of educational material, the disciplines are divided into smaller units, called modules.

The efficiency of scientific and pedagogical potential and quality of training in high schools, to some extent depend on the level of educational process organization. One part of this process in Ukraine - schedule - regulates the labor rhythm, affects the creative impact of teachers, so it can be considered as a factor in optimizing the use of limited human resources - teachers. Since the interests of participants of the educational process are varied, the problem of scheduling is multicriterion.

The scheduling is done by teaching department. This process is unlike in European countries, because students cannot freely choose their subjects of interest, but follow a well-defined schedule, which is approved by the educational council of the institute of
higher education. A new schedule is formed every semester; it consists of 10-15 items. After some of them, a student must pass an exam at the end of the semester.

Ukrainian universities use the traditional 5-point scale:

"5" = "excellent"
"4" = "good"
"3" = "satisfactory"
"2" = "unsatisfactory".

Beginning with 2006 (and in some universities even earlier), students' knowledge is evaluated using the scale from 0 to 100, or rating scale. This can be converted into 5-point scale as follows:

- From 90 (or 91) to 100 = "5"
- From 75 to 89 (or 90) = "4"
- From 60 (or 61) to 74 = "3"
- From 0 to 59 (or 60) = "2".

The expediency of ECTS usage is obvious. Coordinated system of course evaluation makes it possible to build up educational programs using university modules that are agreed upon. Virtual educational environment access requires the ability to measure the informational intensity of courses of different universities.

4.2 Europe system course planning

The needs of higher and secondary vocational education in the individualization of education, reflected in giving the opportunity to build individual educational trajectory, bring up the problem of evaluation methods coordination of educational achievements and their conversion during the transition from one educational program to another. This problem solving involves the development of principles and methods of unified credit system functioning to provide an institutional framework for the educational mobility.

Today, elements of credit-modular organization of vocational education programs are developed and used in a large number of national educational systems; and in connection with Ukraine's accession to the Bologna process, the development and use of credit-modular organization of the educational process has become urgent for our country.

Providing educational mobility by introducing a uniform system of credits (credit units), constructed on basis of modern information infrastructure of data accounting, storage and accumulation on the results of education and training received by students in various educational institutions, as well as dissemination of modular construction principle of program building are to provide an institutional opportunity for learning in different educational institutions within the same individual educational trajectory, built by students on their own. In connection with the introduction of credit-modular system in higher and secondary vocational education, the possibility of educational program selection will expand; education and training will become targeted and more effective.

Course planning is the courses determined by the department in accordance with the level of development of modern science and the traditions of scientific schools, which can enhance, supplement and systematize the theoretical and problematic content of mandatory courses.

Optional courses are offered by the faculties, taking into account the sequence logic of courses in each of the proposed cycle. Optional courses are opened, depending on the number of students who choose this course (minimum 10 students). According to the current curriculum, each student at the beginning of each semester makes individual training plan, following these parameters.

During bachelor degree program student must complete all the training programs provided by the educational program, the total should be evaluated in at least 240
credits (credit units) for bachelors, 300 credits for associates, and during master’s program the student should have 120 credits.

The number of credits or the number of credit units is a conventional numerical index of all laboriousness assigned to individual training course and other forms of educational work (course work, degree work, etc.). It determines the course contribution to the overall student academic load. In other words, it is a unit of measurement of the total labour expenditure of a student to perform a particular work (theoretical and practical training, course and degree works, etc.). One credit is 36 hours of educational time.

Educational time consists of several positions:

- Teaching load, which is about 50% of the total study time (lectures, seminars, practical and laboratory classes in foreign languages, and group academic tutorials).
- Extracurricular load:
  a. Time spent on independent work of students for the learning of the course:
     1. Time required to read the recommended literature and making a summary of the studied courses.
     2. Time required for preparation of written works and oral presentations;
     3. Time needed to prepare for the tests and other forms of intermediate and final assessment. Assessment is a form of written verification of students' knowledge of semester educational program.
     4. Time provided for making a feature, writing articles (for journalists).
     5. Time to prepare for practical and laboratory classes in foreign languages.
  b. Time provided by teaching plan for a practical training, the preparation of course and diploma works, master's theses, final qualifying works, preparation for state exams, etc.

During the year of education in accordance with the basic educational program, the student must have at least 60 education credits. In accordance with the schedule of the educational process in credits, a student in odd semester (1, 3, 5, 7) must attend courses totalling 27-30 credits, in even semesters (2, 4, 6) must also attend courses totalling 27-30 credits, and prepare course work, which is estimated to number of credits, specified by the educational plan for each of the areas and specialties.

Standard load requires approximately 40 hours per week, of which 8-12 hours are instructor led, the rest set aside for independent work in the library and writing essays. Exceeding number of units in a semester is not a violation.

The student draws up individual educational plan on basis of the current curriculum, and contains a set of courses in this speciality taking into account all possible further specializations in master's degree and postgraduate course.

To compile "Individual Student Plan" in accordance with the schedule of the educational process the student has the opportunity to read the description of the courses and consult with teachers from their departments and tutors and draw their individual plan with their help. The procedure and rules for courses selection are determined on basis of the current curriculum according to the educational process. In case of difficulty in courses selection on their own, the students are trained on basic plan (recommended by faculty or tutor).

Courses selection is completed after two start-up weeks before the holidays. "Individual student curriculum" is signed by the tutor and the dean of the faculty during the last three days before the holidays and sent to tutors’ room. After its signing, the plan is mandatory and can not be changed.

Implementation of the course planning system is one more step towards the European Union for Ukraine. In addition, this system will simplify course planning for other universities that use credit-modular system and the "Individual Student Plan" in their practice.
5 Description of the system's results conceptual, logical and physical design

In order to study the communication of JADE agents on the Jadex platform there had been chosen two agents that are exchanging messages. One agent is "Webbridge agent". This agent enables the agents work through the browser, that standard Jadex features do not allow (Alexander Pokahr and Lars Braubach, 2008). The main goal within the Webbridge is to support the communications of a user with the applied agents. Another agent is "agent of course planning", which allows choosing the courses based on the students preferences and takes into account the previously studied courses, and using the capabilities of agent technology. Let us consider the development of model communication between JADE agents on the Jadex platform, which enable to make the cross-platform system of course planning.

5.1 Short description of development
Development and research of intelligent JADE agents on Jadex platform communication is the main objective of this thesis. On the one hand we need to understand the peculiarities of intelligent agents development and on the other to create a multi-agent system on the agent platform. Agent system is a platform that can create, interpret, run, move, and delete agents. After reviewing a number of agent platforms FIPA-compliant JADE platform was chosen and Jadex environment that provides a basic set of functions supporting the process of constructing intelligent agents was created on it. It represents a conservative approach to the agents. 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, ie in Java. J# provides the basic structure and set of programming tools that facilitate agents creation and testing.

These agents will be constructed for the system of courses selection, which is the part of distance education system, so it is necessary to understand the specifics of the agents environment. The system should have a database, which is able to work in a client-server version or as an embedded DBMS with support of "DB in memory" mode. HsqlDB, completely written in Java and retaining all the advantages of SQL, is selected. It can work with data up to 8 GB, supports almost all ANSI92 SQL standards, as well as SQL99 and SQL2003 extensions, but takes only 100 kB RAM.

The architecture is also an important issue. For more information related to the elements of architecture, refer to next part of this chapter.

5.2 Architecture
In the process of agents communication the course planning system should provide the ability to work with agents via browser with a large number of users. Web-based agent application should be built through the interoperability with web browsers and should be manually built-in into agent applications. The generalized web-agent framework allows us to concentrate on the application problem, ignoring the technical details.

Webbridge agent allows combining the capabilities of agent technologies with the capabilities of web applications (Fig. 5.1). Webbridge agent is "glue tier" therefore it allows to determine the distinguishes between details of the planning agent and the web layer. By developing a system using Webbridge agent 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.

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

![Figure 5.1: Communication JADE agents on the Jadex platform in browser](image)

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. Administrator has the right to 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, interesting for the student (subject);
- course level (A, B, C).

The number of selection courses credits should not be less than 30. The number of available places at the course is also taken into account.

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 that are to work in the system must ensure its proper functioning by adding resources of JADE-based agent technologies.

### 5.3 Functional specifications definition

Software development agent-oriented paradigm derives from the object-oriented paradigm and is the product of its evolution. Agent is an elementary building block in the agent-oriented approach. Agents are autonomous and perform for their own internal purposes, given by the developer. Another difference between agents and objects lies in their interacting modes. Objects interact with each other using method invocation, while the agents interact through the Agent Communication Language (ACL), using messages. While using messages for communication agents do not have to know anything about the structure of other agents. The agent has only to send and receive messages from other agents.

Most multi-agent systems developing methodologies are based on the expansion and addition of object-oriented methodology. Now one of the most used methodologies is
Scott A. Deloach and Mark Wood developed methodology (M. Wood, S. Deloach, 2000). This methodology is based on the existing iterative approach to the development of object-oriented systems and fully involves the analysis and design phases for the development of multi-agent systems. Analysis phase includes three steps: goals setting, defining roles and developing of precedents usage. The first step, such as goals setting, is based on the specifications of the system. The result of this stage is hierarchical diagram of system goals.

Consider this stage in the context of Agent of Course Planning. Goals diagram shown in Fig. 5.2:

![Figure 5.2: Goals diagram](image)

On figure 5.2 shows such purposes as «User interaction» (1.1), «User’s information management» (1.2), «Selection courses» (1.3) и «Other goals» (1.4), which do not belong to the Agent of Course Planning, and relate to system as a whole. Goal «User interaction» defines the task of organizing a user's interaction with the system, «User's information management» includes tasks such as forming user profile, modifying user profile, extracting information from the stored profile. The goal «Selection courses» means the courses selection by means of various filters, based on user preferences. «Other goals» includes all other courses planning system purposes, such as work through the Internet, message handling request / response to web-browser, etc.

The next step is to make clear the roles using specific goals and building role diagrams. All roles must include all the goals; thus, our system will achieve the objectives set at the stage of objectives defining. Messaging protocols between the agents are developed at the design stage on basis of interfacing protocols. Designed roles chart depicts Fig. 5.3:
On roles diagram are different roles and tasks that performed by these roles. So from the goals diagram selected following roles: «User interaction», «User’s information management», «Selection courses and «Other roles». Each role had tasks to perform. Therefore, role «User interaction» had the following main tasks: data organization, print data and authentication. Role «User's information management» had such tasks as: to control user's profile, choose filters to courses, search request. For combined role «Other roles» all the other tasks performed by the system were assigned.

The next phase of the analysis stage is to develop precedents of system usage, as well as plotting sequences diagrams for precedents.

For modeling the work of agents we build a UML Use Case diagram (Figure 5.4). It shows three actors: Student / trainee, agent of Course Planning and Webbridge agent. Each of them, perform certain actions and is a link of planning chain.

Figure 5.3: Roles diagram
Therefore, on this phase selected the following precedents: “Start browser and choose link”, “Login”, “Set the parameters”, “Authorized access”, “Add, delete and choose courses”, “Filtering courses by parameters”, “Credits count”, “Vacancies count”, “Manage student’s courses”, “Forwarding web request”, “Forwarding the data to suitable jsp page”, “Send message with request/response data”, “Support type of agent”, “Manage the communicate with agent”, “Support easy BDI structure implementation”, “Configurable session timeout”. Let's take a look the precedents in more detail

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start browser and choose link</td>
<td>describes the operation of program startup and start of work of agents</td>
</tr>
<tr>
<td>Login</td>
<td>describes the operation of user authenticity check at logon. The user enters his/her name, surname, date of birth and password. This data is transmitted to the agent interacting with the user interface and checks for the existence of the user. Extraction data from current user profile in case of successful authentication, and in case of failed authentication - the error message will appear.</td>
</tr>
<tr>
<td>Set the parameters</td>
<td>describes the operation of the filter parameters setting, that are charged with choose courses</td>
</tr>
<tr>
<td>Authorized access</td>
<td>describes the operation of new user registration. The user enters personal data, which are transmitted to the agent responsible for the registration and authentication of users. If the registration is successful, so for registered user there is created a personal profile.</td>
</tr>
<tr>
<td>Add, delete and choose courses</td>
<td>describes the set of operations, that user can do with courses. If the user is administrator, he/she has the possibility to add and delete courses. The user - student can choose courses that are interesting or uninteresting to him/her.</td>
</tr>
<tr>
<td>Filtering courses by parameters</td>
<td>describes the operation of courses selection by parameters. In the system we can select by programs and courses, by key words, by interesting direction(topics) for the student, by level of the course (A, B, C).</td>
</tr>
<tr>
<td>Credits count</td>
<td>describes the operation of counting credit points for selected courses, if their number is less than 30 so courses will not be selected, and the user will see a warning message.</td>
</tr>
<tr>
<td>Vacancies count</td>
<td>describes the operation of counting vacancy by course, if all are in use, so the course will not be selected, and the user will see a warning message.</td>
</tr>
<tr>
<td>Manage student’s courses</td>
<td>describes the set of operations concerning the user courses management.</td>
</tr>
<tr>
<td>Forwarding web request</td>
<td>describes the operation that inform the servlet how to establish the connection, the address of the coordinator agent can be specified in the configuration file of the web application (web.xml).</td>
</tr>
<tr>
<td>Forwarding the data to suitable jsp page</td>
<td>describes the operation of delegate servlet that obtains the result from the coordinator agent and forwards the data to suitable JSP page</td>
</tr>
<tr>
<td>Send message with request/response data</td>
<td>describes the operations of data message exchanging required for the correct operation of the agent systems built on Jadex platform.</td>
</tr>
<tr>
<td>Support type of agent</td>
<td>describes the operation of type agent support that is specified in the web.xml file, if the type is not specified, so there will be used the default one.</td>
</tr>
<tr>
<td>Manage the communicate with agent</td>
<td>describes the operations that are responsible for communication management between the agents. Each such communication includes several messages from the agent which started &quot;talk&quot; to its &quot;interlocutor&quot;, and vice versa. Communication between agents is implemented according to protocols that had developed by FIPA organization</td>
</tr>
<tr>
<td>Support easy BDI structure implementation</td>
<td>describes the operations responsible for the implementation of standard elements of BDI architecture. To select the suitable plans, BDI-style of reasoning is applied, e.g. plans were selected according to pre- or context-conditions and alternative plans are tried to be applied, when some selected plan fails</td>
</tr>
<tr>
<td>Configurable session timeout</td>
<td>describes the operation of session timeout setting.</td>
</tr>
</tbody>
</table>

**Table 5.1: Precedents**
In order to analyze the work of the agents construct a series of diagrams. Sequence Diagram (Fig. 5.5) allows us to trace components active in certain stages. We should note that this diagram contains only components associated with the student work with the system and does not show interaction with other users system, such as the administrator or other agents that must interact with the system. This view of the system, of course, is one-sided, but still allows us to examine system interaction with the user.

![Sequence Diagram](image)

Figure 5.5: Sequence Diagram

Figure 5.5 displays the user’s actions after registration. After the registered user enters, its data is extracted from the database, and the data obtained as the result of planning, is put to the database.

Action diagram allows to view the algorithm of agent of course planning (Fig. 5.6).

Upon completion of this stage, ontology, developed for the communication of agents, is usually modified, and developed agents are implemented.

Interfaces are boundary objects between the user and program procedures; they give the user an indirect access to the program methods and functions performance. To determine the agent in Jadex, it is necessary to formulate its facts (beliefs), goals and plans. To work with the platform Jadex, you need to create two types of files: ADF files (Agent Definition File) - XML file that defines the agent and Java classes to implement agent plans.
The Statechart diagram (Fig. 5.7) displayed processes between agents.

Let us examine the BDI structure of courses planning agent as an example of JADE agent development.

Beliefs:
1) user_name – user name (belief);
2) user_password – user password (belief);
3) user_id – user identification number (belief);
4) max_credits – default credit points equal 30 (basic knowledge).

Basic knowledge is a permanent part of the agent knowledge about itself, about the environment, as well as permanent knowledge about other agents, that do not change within the life cycle of the agent. Beliefs are the variable part of the agent knowledge.
about the environment and other agents that may change over time, but the agent may not realize this and continue to use it for its purposes.

Goals:
1) web_request – forward request data to webbridge agent;
2) different goals to show page.

The presence of the agent goal formation mechanism provides a fundamentally new autonomy level. The agent is not obliged to follow instructions of another agent or user; it just depends on environmental conditions, including the goals and intentions of other agents. By contrast with the object, the agent can take on certain obligations or, alternatively, refuse to perform some tasks, reasoning the lack of competence, being busy by another task, and so on. At the same time, the agent can perform such actions as generation, suppression and substitution of other agents, functions activation (both own and other agents), action script activation, other agents current state storage, etc.

Plans:
1) loginUser_plan – to check the data correctness;
2) registration_plan – to load the selected data for registration and make a goal to show on registration page;
3) filter_plan – to make a goal to show page with filters (parameters for choice courses);
4) signin_plan – to add data to data base and make a goal to show page with filters after registration;
5) middle_plan – to make a list of available courses (considering filters) and make a goal to show courses available for user;
6) main – checking of the data correctness and data transferring to the proper page;
7) addcourses_plan – add/extract courses into/from data base and make a goal to show page of course list;
8) bad_course_plan – add/extract courses into/from data base and make a goal to show page of uninterested course for student;
9) start_course_plan – start or finish course;
10) list_course_plan – extract list course from data base and make a goal to show page for all courses (available for administrator);
11) delete_course_plan – delete the course (available for administrator);
12) add_new_course_plan – add the new course to data base (available for administrator);
13) insert_course_plan – make a goal to show page for new course adding (available for administrator).

Agent needs plans to work in an environment. With the help of certain expressions (or without them) the events are handled in the body of a plan; triggers, parameters, messages or purposes can be used here.

You can see designed XML model in Appendix A, it uses the BDI architecture, and allows you to simulate the agent of course planning. All this clearly indicates that the agent, as «active object» or «artificial agent», forms its own behavior, and it is at a higher level of complexity as compared with the traditional objects in the OOP.

5.4 Research of communication Jade agents on the Jadex platform
As the example of course planning agents communication we can study the behavior of agents on Jadex platform. Figure 5.8. shows facts, line advance to the browser, and various interagent messages as input information, as output we have achieved goals; everything is controlled by FIPA protocols and managed by agents community.
In the process of work, each block performs a certain plan previously described in the structure of a particular agent. The implementation of it in some cases leads to the achievement of certain goals.

5.5 The Data Base structure

The system uses the database storing the data about the user, the list of courses attached to educational program, school, level and other parameters that are considered in Table 5.2.

<table>
<thead>
<tr>
<th>№</th>
<th>Name</th>
<th>Description</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>courses</td>
<td>All possible courses</td>
<td>ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Identification number</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>code</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Code of course</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>credit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Number of credit points</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Name course</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>max_num_students</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Number of students per course</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>free_places</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Number of vacancy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>keywords</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Keywords of course</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>topics_id</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Topic of course</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>view_study_id</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Full semester/Simple program</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>studprogram_id</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Study program</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>studprogram_school_id</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>School</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>level_id</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Level</td>
</tr>
<tr>
<td></td>
<td>school</td>
<td>All possible schools</td>
<td>ID</td>
</tr>
<tr>
<td>---</td>
<td>--------</td>
<td>----------------------</td>
<td>----</td>
</tr>
<tr>
<td>3</td>
<td>studprogramms</td>
<td>All possible study programs</td>
<td>ID</td>
</tr>
<tr>
<td>4</td>
<td>level</td>
<td>First Level/Second Level</td>
<td>ID</td>
</tr>
<tr>
<td>5</td>
<td>topics</td>
<td>All possible topics</td>
<td>ID</td>
</tr>
<tr>
<td>6</td>
<td>view_study</td>
<td>Full year / Full semester Simple program</td>
<td>ID</td>
</tr>
<tr>
<td>7</td>
<td>country</td>
<td>List of country</td>
<td>ID</td>
</tr>
<tr>
<td>8</td>
<td>bad_courses</td>
<td>Uninteresting courses for user</td>
<td>ID</td>
</tr>
<tr>
<td>9</td>
<td>students</td>
<td>Personal user data</td>
<td>ID</td>
</tr>
</tbody>
</table>
Conceptual database schema is shown in Figure 5.9.

The EER (Enhanced Entity-relationship) model includes all the modeling concepts introduced by the ER (Entity-relationship) model. It is an ontology in area of interest. The database consist of 10 data tables related using key field. The following features are implemented in software:
- Data saving in text format.
- Data removing and changing.
- Maintenance of data referential structures.
- Branching by links.
- Sampling and sorting by different sets of fields.

The logical structure of the database is defined by peculiarity of automation object. The focal point of the system is the student. Student data and courses management are implemented by means of agents. Detailed data, put during the system operation, is a set of ancillary entities. These include the topics, the level of student, specialty, department, type of education, courses, countries that can be created by the administrator.
5.6 Summary

A model of multi-agent system, which focuses on the user and reacts to its actions, allowing you to improve the quality of education, is proposed. Also agents that are sufficient to demonstrate the system were created. They are self-learning and solve interagent conflicts, so they do not require the constant online presence of the teacher. Agents exist, exchange information, and interact in Jadex environment. Agents are built on the basis of the deliberative architecture, in which the agent is regarded as a "black box", sending and receiving messages. Each agent has a unique set of goals, facts, plans, and other components that are sufficient for its proper functioning. The system has a database that stores user data, the data obtained in the process of the communication with the system, and also a list of courses and other tables taking part in the process of courses selection.
Software implementation of model communication between JADE agents on the Jadex platform

In this Chapter we describe a main requirements of system and communications and user communication with the system. The requirements describe the characteristics of computer to system correct work. And user communication are for more understandable explanation of work in the system.

6.1 Equipment Environment
Minimum system requirements for correction work of system

<table>
<thead>
<tr>
<th>Platforms</th>
<th>All Platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browsers</td>
<td>All Browsers Mozilla 1.4.x or 1.7+, Firefox 1.06+, IE 6.0 SP1+ or 7.0</td>
</tr>
<tr>
<td>Random Access Memory (RAM)</td>
<td>64 Mb</td>
</tr>
<tr>
<td>Disc Space</td>
<td>110 Mb</td>
</tr>
<tr>
<td>Java Virtual Machine</td>
<td>1.6 and more</td>
</tr>
</tbody>
</table>

Table 6.1: System requirements

6.2 Software Environment
Software operates on Jadex platform that supports JADE agents development. To build JADE agents there are used 2 type of files. There are ADF and Java-class files. The ADF file is file of XML format. There are described BDI components such as goals, plans, beliefs (facts) and other there. There are also described FIPA protocols. Each of plan is linked with Java class that describes different mechanisms of agent work.

For easy portability of program we use HSQLdb database management system. We also use ORM Hibernate that can provide changing part of code of hibernate properties and do the system supported by MySQL database management system. Program works in browser. We use Java, HTML, JavaScript for jsp-pages, and CSS for appearance.

6.3 Communications Requirements
Communication requirements are displaying the external and internal communications between program modules.

6.3.1 Communications Overview
Graphic expression showing the known communication requirements of the system is described in Chapter 5. There are UML diagrams such as Use Case, Sequence, Action, as well as diagrams IDEF0 and EER.

6.3.2 Communications Software
System being developed includes such software as:

<table>
<thead>
<tr>
<th>Languages</th>
<th>Java, HTML, XML, CSS, JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database management systems</td>
<td>HSQLdb</td>
</tr>
<tr>
<td>Agent platform</td>
<td>Jadex</td>
</tr>
<tr>
<td>Agent environment</td>
<td>JADE</td>
</tr>
<tr>
<td>Methodological approach</td>
<td>MSF Process Model. Spiral model</td>
</tr>
</tbody>
</table>
6.4 Interfaces
The process of work with system includes few steps.
1) modules of agent connection;
2) Internet connection;
3) login;
4) select filters;
5) choice courses;
6) save results.

6.4.1 Modules of agent connection
To start system:
1) Start server (e.g. Tomcat).
2) Start the “start.bat” (that starts Jadex).
3) Add jadex-web-bridge.jar (Webbridge agent).
4) Add CoursesAgent.jar.
All process display on Figure 6.1.

Figure 6.1: Jadex with started agents

The left bottom window displays active agents that started on platforms. In our case these are Coordinator (it is Webbridge agent) and Courses1 (it is our agent of course planning)

6.4.2 Work with agents
The system is web application. It starts in browser. After starting browser we can see main form for login (Fig. 6.2). It is communication between two agents.
Agent of course planning controls entering of student to the system. To enter the system the user types his/her surname, name and birthday date. On the basis of these data the system generates unique name (nickname). Nickname and password are compared with data in database. If data are correct, so the agent will obtain the personal profile about the otherwise it obtains an error massage about registration or data incorrectness.

In case if user wants to register, the agent will display the following form (Fig. 6.3).

Agent controls data entry and shows the warning messages. If the data are correct, all user data will be saved in database. Nickname and password are the facts in knowledge base.

After user logged in the system there should be chosen the parameters for course selection. For this purpose we use filters (Fig. 6.4).
The filters allow agent to understand the student preferences. Student can alter any filters after login. For the convenience we create the lists “Study Program”, “Level”, “Interesting Topics”.

After that the agent generates list of courses which could be interesting for student (Fig. 6.5).

Clicking on any course we can see information and the students' comments about each of them. The example of information about course is shown in Figure 6.6.
On page with available courses it is possible to transfer the courses to the lists of uninteresting courses and to select courses for the study. He/She selects courses and options to be added to any list. The agent analyzes the courses and moves them to the desired list. If the settings for course adding are incorrect, so the user will see a warning message.

The page with uninteresting (or “bad”) courses is shown in Fig. 6.7. Uninteresting courses will not be offered further to the user, but he/she has the opportunity to view and to restore them.

The page with example of error massage is shown in Fig. 6.8. The error message with different data is shown to user when some data are invalid.
The page “List selected course” is shown in Fig. 6.9. This list of courses is interesting for student and he/she selected them for study.

If student starts course learning the course is disabled for any action. If user marks that course is finished the agent analyzes the data and if it is possible the course is considered as learned, certainly if this course is of the same level. The student shall take the other available course from this list. This process is shown in Figure 6.10.

As a result of the work done we can insist that the intelligent agents are very prospective direction that could help people to get out of the existing lack of
information. Personal agents begin to apply in such area as distance education being very important tool to facilitate the work of the teacher. They will make education more qualitative and personalized. They focus on the human senses and allow the student to build his/her own educational program.

Agents can send messages to other agents, change the base of belief, create new goals and cause internal events. In our model, agents represent the individual objects of education. Each of them has clearly identified functions. They share their knowledge in environment through messages. The established model of agents’ communication allows building a system that facilitates the course planning and enables the researching of agents’ behavior. It will allow building a fully functional multi agent educational system, which will operate efficiently due to the optimal model of agent communication.
7. Conclusions and Future Challenges

The main topic of this thesis focuses around one fundamental principle of agent-oriented system. Thus, in this chapter we discuss the application, concerning the communication between JADE agents on the Jadex platform. Moreover, we give a description of different aspects of the research, based on agents, and we present the multi agent system for the remote education.

7.1 Results

In the developed system the JADE agents operate on Jadex platform, they use the principles of agent-oriented programming. A key advantage of agent-oriented programming is the ability to manage the uncertainty of supply and demand, as well as numerous changes.

To achieve the main goal of our work we chose two agents that are most closely communicate together and which are sufficient to demonstrate the agents' work in the system of remote education. These are webbridge agent and agent of course planning. The system is complex, because it consists of two independent components that communicate actively with each other. The easy access is realized through the standard browser. In our system the behavior of agents is also described in terms of beliefs, desires and intentions (main BDI principles), which is based on the concepts from sociology, and the communication is defined in protocols terms, that don’t have direct coupling with the first one.

The BDI allows us to build the original model of agents’ behavior and describe the step-by-step response to different events in the system. Agent operating according to certain plans obtains the concrete goals. JADE is one of few platforms that based on FIPA specification. The FIPA protocols of communication allows agent to reduce the search area of possible solutions. It defines the limited range of responses to the messages that is possible for this situation.

During the work we have completed the set of following goals such as:
1. Analysis of methods for building agent-oriented systems as presented in Chapter 2 and there had chosen the most suitable one.
2. Analysis of agent architectures as presented in Chapter 3. And the deliberative agent architecture had chosen as the most suitable one for the effective agents’ communication in the course planning system.
3. Analysis of specific systems of education and systems of course planning as presented in Chapter 4.
4. There were selected agents required for the proper operating of the course planning system and the system architecture was built (Chapter 5).
5. There were developed the model of process in the system, the diagrams such as goals, roles, Use Case, Sequence, Action, that show the system operating process and method of communication between the agents (Chapter 5).
6. There were distinguished goals, facts and plans (BDI schema) of agents, motivations and schemes of agents’ cooperation (Chapter 5).
7. We can change system by expanding of existing objects or adding new features (Chapter 6).
8. There was developed the user-friendly database, which is easy to transfer and easy to use (Chapter 5).

The purpose was to show some results:
- There was created model (IDEF0) that shows the communication of agents in multi-agent system on Jadex platform. These are communications using the JADE agents’ capabilities.
There was created software complex of course planning with two JADE agents and all features intrinsic to them (Chapter 6).

### 7.1.1 Reflections

Every semester the students realize the course planning. Many universities have found the whole list of steps for course planning. It would be much more convenient if the procedure were controlled by a certain person. But intelligent agents could replace it. They have many advantages and they are able to analyze the user behavior, and to assist him/her in the planning process. The agents can automate the process of course planning. The students will be able to read the comments of other students about the course or a lecturer, and they will view a brief description course programs. Besides, it gives an advantage during course choosing by setting the parameters of interest. You can also find students‘ courses that had been studied earlier in the education process. The Course Planning will help students to become intentional, responsible and able to study.

This thesis describes two different areas. There are researched the agents’ communications on the Jadex platform and development of course planning system. Researching the first area, we built a model of communication between agents, which is based on the exchange of FIPA messages and on the plans to achieve the goals. Other area simplifies the course planning for students.

These applications are used to research the main conception of this thesis – research of JADE agents communication as agent-oriented technology. We could pay more attention to the following matters:

1. To add timetable to each course.
2. To divide into the semesters.
3. Optimization - achieving of the most efficient design for the software system.

### 7.2 Future Challenges

Agent should, on equal terms with a person, to participate both in problem definition and in the algorithm selection for its solution and implementation in a fixed world. Nowadays there are actively used the systems of remote education. Now there were developed many systems for remote education, but none of them can ensure its optimal coverage, which provides communication between the student and teacher. To organize and support remote education process we suggest developing of multi agent system, which consists of a set of educational objects. Educational objects are agents. The system can consist of the following groups of agents:

1. Agents of commutation: the agent coordinator, agent of course planning, chat agent, personal agent;

This stated agents is not limited to just this set of proposed agents. It can be expanded based on different needs in expanding of multi agent system functionality of remote education.

Student does not attach to a teacher's structure and sequence of the course passing which is set by teacher. And he/she has opportunity to study the material in accordance with his/her personal preferences, abilities, level of material mastering, previous experience of use of the remote education system, and also basic knowledge.

Work with an agent in multi-agent system of the remote education is as following. Student uses a model that is focused on the implementation of agent-oriented approach in developing of educational systems. The set of agents – objects of education and a logical channel with messages build the course passing trajectory for the student.
Every moment the student has access to only one agent of remote education and follows the commands generated by the agent depending on its facts, goals and plans. When certain conditions are achieved, the agent sends a message to another agent with different objectives, for example, to make information request or redirect the student to the next part of the educational program.

Transitions and communications of agent with other agents form the preferred educational route for the student. This helps student to avoid the static sequence, which teacher is submitted to. The sequence of transitions is controlled by the agent coordinator. The main goal of the agent is to achieve the optimal route and to indicate the relatively easy way of studying for student.

It is reasonable to use JADE environment to build agents of multi-agent remote education system. Its main goal is to create and to distribute the multi-agent systems and applications that meet the FIPA standards of intelligent agents. The Jadex platform could also be used, this will provide a platform-independent set of agents. They communicate in the same logical channel. Since the system supports the personalized customer communication with educational system, helps in the automatic evaluation and provides development tools, which are easy to use, of interactive multimedia applications, and actively communicates with the user, so it requires the powerful reliable means of implementation.

The intelligent agents can provide a process of active remote education communicating with each other. In this case, multi agent system is self-training in concordance with the user’s response to some events during the course makes its own recommendations and hints. It makes possible to reduce the work of a teacher turning him into a tutor. In this system, the teacher needs only qualitatively prepared teaching material, and intelligent agents make the maintenance of the whole educational process.
7.3 Conclusions
The urgency of the theme is conditional on growth of the number of people interested in multi agent technologies. Nowadays there are large quantity of multiform architectures and types of agents. Because of it there appears complication of agents’ selection for certain software problem. The lack of the standards of architecture using for the certain type of task also causes the problems with development of multi agent systems, which nowadays starts to apply in education.

Using agents should qualitative improve the educational process, especially in self-training systems. They can adapt to every student and enable the maximum approximation to the real model of education. Agents will enable the automation of course planning process. This will manage student's time.

On the basis of requirements to the development of software environment that is formulated in developed enterprise model and in the tasks list, Jadex was selected as the development environment which allows agents building on the basis of BDI architecture. The open source of development – JADE was also used. The FIPA protocols are to support the communications between agents. All the technologies are using the development environment Java, so the built system is platform independent.

The main functions of developed program module are:
- User authorization.
- The response to user actions (show prompt, alert massages, check points and use these data on-stream).
- The selection of courses, that fit user's study program and his/her level of education.
- The selection of courses by keywords.
- The selection of courses according to course level.
- Courses have the constraints on maximum number of students per course, and contain the information about quantity of students that are already registered for some courses.
- Categorizing the courses into a number of subjects
- Agent allows the student to choose which of subjects are interesting for him/her.
- The student could state the courses that are not interesting for him/her, and the agent will re-plan without including data of uninteresting courses.
- The sum of chosen courses is not less or equal to 30 credit points.
- The amount of courses is not over 4.
- The analysis of results.
- The data storage in database.

Thereby, as the result of the assigned tasks solution we researched the communication of JADE agents in the Jadex platform. And it allowed us to affirm that suggested methods of agents’ implementation enable to build a system of remote education, which is effective due to the suggested model of agents’ communication.
Glossary

ADF – Agent Definition File
ADL – Anti-Defamation League
AICC – Aviation Industry CBT Committee
AMS – Agent Management Service
API – Application Program Interfaces
ACC – Agent Communication Channel
BDI – Belief – Desire – Intention
DF – Directory Facilitator
EER – Enhanced Entity-relationship
FIPA – Foundation for Intelligent Physical Agents
GUI – Graphical User Interface
IDE – Integrated Development Environment
IDEF - Integrated DEFinition
IEEE – Institute of Electrical and Electronics Engineers
IMS – Intelligent Manufacturing System
IA – Intelligent Agents
LTSC – Learning Technology Standards Committee
MAS – Multivalent system
MVC – Model - View - Controller
RAM – Random-access memory
RPC - Remote Procedure Call
UML – Unified Modeling Language
JADE – Java Agent Development Framework
XML – eXtensible Markup Language
References


Brockhaus F.A., Efron I.A (1890) Entsiklopedicheskii slovar


Internet sites

http://shcherbak.net/, viewed 2009-09-17
http://jade.cselt.it/, viewed 2009-09-17
http://www.acm.org/crossroads/xrds5-4/multiagent.html, viewed 2009-09-17
http://www.aaai.org/AITopics/pmwiki/pmwiki.php, viewed 2009-09-17
http://aivanoff.blogspot.com/, viewed 2009-09-17
http://systech.miem.edu.ru/, viewed 2009-09-17
http://www.fipa.org/, viewed 2009-09-17
Appendices

Appendix A  ADF file of agent of course planing

<!--

<H3>The PlaningCourses agent.</H3>
It has the task to help student planing they courses follow year.
-->

<agent 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-0.96.xsd"
name="Courses"
package="jadex.bridge.courses.planingAgent">

<imports>
  <import>jadex.bridge.courses.DB.*</import>
  <import>jadex.*</import>
  <import>jadex.planlib.*</import>
  <import>jadex.runtime.*</import>
  <import>java.util.logging.Level</import>
  <import>jadex.bridge.onto.*</import>
  <import>jadex.bridge.application.*</import>
</imports>

<capabilities>
  <capability name="webcap" file="jadex.bridge.application.WebInteraction"/>
</capabilities>

<beliefs>
  <belief name="user_name" class="String"/>
  <belief name="user_password" class="String"/>
  <belief name="user_id" class="Integer"/>

  <belief name="max_credits" class="Double">
    <fact>30.0</fact>
  </belief>
</beliefs>

<goals>
  <achievegoal name="web_request">
    <assignto ref="webcap.web_request"/>
    <parameter name="request" class="jadex.bridge.onto.Request">
      <assignto ref="webcap.web_request.request"/>
    </parameter>
    <parameter name="response" class="jadex.bridge.onto.Response"
    direction="out">
      <assignto ref="webcap.web_request.response"/>
    </parameter>
    <parameter name="type" class="String">
      <assignto ref="webcap.web_request.type"/>
    </parameter>
  </achievegoal>
</goals>

51
<goals>

<plans>

<!--Show main page-->
<plan name="loginUser_plan">
    <parameter name="error" class="String">
        <value>$goal.request.getParameterValue("error")</value>
    </parameter>
    <body>new LoginPlan()</body>
    <trigger>
        <goal ref="web_request">
            <match>"index".equals($goal.type)</match>
        </goal>
    </trigger>
</plan>

<!--Show registration page-->
<plan name="registration_plan">
    <parameter name="error" class="String">
        <value>$goal.request.getParameterValue("error")</value>
    </parameter>
    <body>new SigninPlan()</body>
    <trigger>
        <goal ref="web_request">
            <match>"register".equals($goal.type)</match>
        </goal>
    </trigger>
</plan>

<!--Show page with filters (parameters for choice courses)--> 
<plan name="filter_plan">
    <parameter name="error" class="String">
        <value>$goal.request.getParameterValue("error")</value>
    </parameter>
    <body>new FilterPlan()</body>
    <trigger>
        <goal ref="web_request">
            <match>"signin".equals($goal.type)</match>
        </goal>
    </trigger>
</plan>

<!--Show page with filters after registrating (parameters for choice courses)--> 
<plan name="signin_plan">
    <parameter name="famName" class="String">
        <value>$goal.request.getParameterValue("famName")</value>
    </parameter>
    <parameter name="firstName" class="String">
        <value>$goal.request.getParameterValue("firstName")</value>
    </parameter>
    <parameter name="userpassword" class="String">
        <value>$goal.request.getParameterValue("userpassword")</value>
    </parameter>
</plan>

</goals>
<parameter name="street" class="String">
  <value>$goal.request.getParameterValue("street")</value>
</parameter>

<parameter name="town" class="String">
  <value>$goal.request.getParameterValue("town")</value>
</parameter>

<parameter name="birthday" class="String">
  <value>$goal.request.getParameterValue("birthday")</value>
</parameter>

<parameter name="email" class="String">
  <value>$goal.request.getParameterValue("email")</value>
</parameter>

<parameter name="postCode" class="String">
  <value>$goal.request.getParameterValue("postCode")</value>
</parameter>

<parameter name="mycountry" class="String">
  <value>$goal.request.getParameterValue("mycountry")</value>
</parameter>

<body>new RegisterPlan()</body>
<trigger>
  <goal ref="web_request">
    <match>"filter".equals($goal.type)</match>
  </goal>
</trigger>

<!--Show available courses for user-->
<plan name="middle_plan">
  <parameter name="programm" class="String">
    <value>$goal.request.getParameterValue("programm")</value>
  </parameter>
  
  <parameter name="level" class="String">
    <value>$goal.request.getParameterValue("mylevel")</value>
  </parameter>
  
  <parameter name="credits" class="String">
    <value>$goal.request.getParameterValue("credits")</value>
  </parameter>
  
  <parameter name="topics" class="String">
    <value>$goal.request.getParameterValue("topics")</value>
  </parameter>
  
  <parameter name="keywords" class="String">
    <value>$goal.request.getParameterValue("keywords")</value>
  </parameter>
  
  <body>new SelectionPlan()</body>
<trigger>
  <goal ref="web_request">
    <match>"middle".equals($goal.type)</match>
  </goal>
</trigger>

</plan>

<!--Checking the correct data and transport data in necessary page-->
<plan name="main">
  <parameter name="famName" class="String">
    <value>$goal.request.getParameterValue("famName")</value>
  </parameter>
  <parameter name="firstName" class="String">
    <value>$goal.request.getParameterValue("firstName")</value>
  </parameter>
  <parameter name="birthday" class="String">
    <value>$goal.request.getParameterValue("birthday")</value>
  </parameter>
  <parameter name="userpassword" class="String">
    <value>$goal.request.getParameterValue("userpassword")</value>
  </parameter>
  <parameter name="error" class="String">
    <value>$goal.request.getParameterValue("error")</value>
  </parameter>
  <body>new MainPagePlan()</body>
  <trigger>
    <goal ref="web_request">
      <match>"login".equals($goal.type)</match>
    </goal>
  </trigger>
</plan>

<!-- Show page to choice courses -->
<plan name="addcourses_plan">
  <parameter name="error" class="String">
    <value>$goal.request.getParameterValue("error")</value>
  </parameter>
  <parameter name="operation" class="String">
    <value>$goal.request.getParameterValue("operation")</value>
  </parameter>
  <parameter name="checkedList" class="String">
    <value>$goal.request.getParameterValue("checkedList")</value>
  </parameter>
  <body>new AddCoursesPlan()</body>
  <trigger>
    <goal ref="web_request">
      <match>"addcourses".equals($goal.type)</match>
    </goal>
  </trigger>
</plan>

<!-- Uninterest course for student -->
<plan name="bad_course_plan">
  <parameter name="checkedList" class="String">
    <value>$goal.request.getParameterValue("checkedList")</value>
  </parameter>
  <body>new BadCoursesPlan()</body>
  <trigger>
    <goal ref="web_request">
      <match>"bad".equals($goal.type)</match>
    </goal>
  </trigger>
</plan>
<plan name="start_course_plan">
  <parameter name="course" class="String">
    <value>$goal.request.getParameterValue("course")</value>
  </parameter>
  <body>new StartPlan()</body>
  <trigger>
    <goal ref="web_request">
      <match>"start".equals($goal.type)</match>
    </goal>
  </trigger>
</plan>

<!-- For ROOT user (nickname=11111999kk, password=111111) -->
<!-- Show page to all courses -->
<plan name="list_course_plan">
  <parameter name="checkedList" class="String">
    <value>$goal.request.getParameterValue("checkedList")</value>
  </parameter>
  <body>new ChangeCoursePlan()</body>
  <trigger>
    <goal ref="web_request">
      <match>"listcourse".equals($goal.type)</match>
    </goal>
  </trigger>
</plan>

<!-- Delete courses -->
<plan name="delete_course_plan">
  <parameter name="checkedList" class="String">
    <value>$goal.request.getParameterValue("checkedList")</value>
  </parameter>
  <body>new DeletePlan()</body>
  <trigger>
    <goal ref="web_request">
      <match>"delete".equals($goal.type)</match>
    </goal>
  </trigger>
</plan>

<!-- Add the new course -->
<plan name="add_new_course_plan">
  <parameter name="code" class="String">
    <value>$goal.request.getParameterValue("code")</value>
  </parameter>
  <parameter name="name" class="String">
    <value>$goal.request.getParameterValue("name")</value>
  </parameter>
  <parameter name="max_places" class="String">
    <value>$goal.request.getParameterValue("max_places")</value>
  </parameter>
</plan>
<parameter name="free_places" class="String">
  <value>$goal.request.getParameterValue("free_places")</value>
</parameter>

<parameter name="programm" class="String">
  <value>$goal.request.getParameterValue("programm")</value>
</parameter>

<parameter name="level" class="String">
  <value>$goal.request.getParameterValue("mylevel")</value>
</parameter>

<parameter name="credits" class="String">
  <value>$goal.request.getParameterValue("credits")</value>
</parameter>

<parameter name="topics" class="String">
  <value>$goal.request.getParameterValue("topics")</value>
</parameter>

<parameter name="keywords" class="String">
  <value>$goal.request.getParameterValue("keywords")</value>
</parameter>

<body>new NewCoursePlan()</body>
<trigger>
  <goal ref="web_request">
    <match>"fake".equals($goal.type)</match>
  </goal>
</trigger>
</plan>

<!-- Show page to add the new course -->
<plan name="insert_course_plan">
  <parameter name="error" class="String">
    <value>$goal.request.getParameterValue("error")</value>
  </parameter>
  <body>new InsertCoursePlan()</body>
  <trigger>
    <goal ref="web_request">
      <match>"insertCourses".equals($goal.type)</match>
    </goal>
  </trigger>
</plan>
</plans>
</agent>
Appendix B  Code of some classes

SelectionPlan.java

package jadex.bridge.courses.planingAgent;

import jadex.bridge.courses.planingAgent.beans.Filters;
import jadex.bridge.courses.DB.tables.CoursesDB;
import jadex.bridge.courses.DB.tables.StudentsDB;
import jadex.bridge.courses.DB.WorkDB;
import jadex.bridge.onto.Request;
import jadex.bridge.onto.Response;

import java.util.List;

public class SelectionPlan extends AbstractWebPlan {
    WorkDB workDB = new WorkDB();
    Filters studentUser = new Filters();

    protected void processRequest(Request request, Response response) {
        String name = (String) getBeliefbase().getBelief("user_name").getFact();
        String password = (String) getBeliefbase().getBelief("user_password").getFact();

        String programm = (String) getParameter("programm").getValue();
        String level = (String) getParameter("level").getValue();
        String credits = (String) getParameter("credits").getValue();
        String keywords = (String) getParameter("keywords").getValue();
        String topics = (String) getParameter("topics").getValue();

        studentUser.setNickname(name);
        studentUser.setPassword(password);
        studentUser.setId_stud_prog(Integer.parseInt(programm));
        studentUser.setLevel(Integer.parseInt(level));
        if (credits == null || credits.equals("") ) credits = "0";
        studentUser.setCredits(Integer.parseInt(credits));
        if (keywords == null || keywords.equals("") ) keywords = "";
        studentUser.setKeywords(keywords);
        studentUser.setTopics(Integer.parseInt(topics));

        response.addParameterValue("filter", studentUser);

        workDB.filterCourses(studentUser);

        final StudentsDB user = workDB.getUser(studentUser.getNickname(),
        studentUser.getPassword());
        List<CoursesDB> dbList = workDB.getCurses(user);
        response.addParameterValue("name_user", user.getFirst_name() + " " +
        user.getFamily_name());
        response.addParameterValue("courses", dbList);
        response.setViewPage("middle");
    }
}

57
public class RegisterPlan extends AbstractWebPlan {

    WorkDB workDB = new WorkDB();

    protected void processRequest(Request request, Response response) {
        Student studentUser = new Student();

        String famName = (String) getParameter("famName").getValue();
        String firstName = (String) getParameter("firstName").getValue();
        String birthday = (String) getParameter("birthday").getValue();

        String nickname = birthday.replace("/", ".") + famName.charAt(0) + firstName.charAt(0);
        String password = (String) getParameter("userpassword").getValue();

        getBeliefbase().getBelief("user_name").setFact(nickname);
        getBeliefbase().getBelief("user_password").setFact(password);

        if (workDB.correctName(nickname).size() == 0) {
            String street = (String) getParameter("street").getValue();
            String town = (String) getParameter("town").getValue();
            String email = (String) getParameter("email").getValue();
            String postCode = (String) getParameter("postCode").getValue();
            String country = (String) getParameter("mycountry").getValue();

            studentUser.setFamily_name(famName);
            studentUser.setFirst_name(firstName);
            studentUser.setBirth(birthday);
            studentUser.setNickname(nickname);
            studentUser.setPassword(password);
            studentUser.setStreet_address(street);
            studentUser.setTown(town);
            studentUser.setEmail(email);
            studentUser.setPost_code(postCode);
            studentUser.setCountry(Integer.parseInt(country));

            workDB.addUser(studentUser);
            getBeliefbase().getBelief("user_id").setFact(workDB.getUser(nickname, password).getId());
        }

        List<StudProgramms> studyProgramm = new WorkDB().getListProgram();
        List<ProgrammList> list = new ArrayList<ProgrammList>();

        for (StudProgramms st : studyProgramm) {
            ProgrammList programmList = new ProgrammList();
            programmList.setId(st.getId());
            programmList.setName(st.getName());
            list.add(programmList);
        }

        response.addParameterValue("listProgramm", list);
    }
}

RegisterPlan.java
List<Level> levelList = new WorkDB().getListLevel();
response.addParameterValue("level", levelList);

List<Topics> topicsList = new WorkDB().getListTopics();
response.addParameterValue("topics", topicsList);

response.setViewPage("filter");

} else {
    response.addParameterValue("error", "This NickName is in the system. Select correct parametrs!");
    response.setViewPage("register");
}

}